UNIVERZA V LJUBLJANI

FAKLUTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Krevl

# Modeliranje aplikacij Interneta stvari

MAGISTRSKO DELO

MAGISTRSKI PROGRAM
INFORMACIJSKI SISTEMI IN ODLOČANJE

MENTORICA: doc. dr. Mojca Ciglarič

Ljubljana, 2016

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Večna pot 113*
*1000 Ljubljana, Slovenija*
*telefon: 01 47 98 100*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Številka: 170-MAG-ISO/2016
Datum: 06. 04. 2016

**Andrej KREVL**, univ. dipl. inž. rač. in inf.

**Ljubljana**

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Modeliranje aplikacij Interneta stvari**

**Modeling of IoT applications**

Tematika naloge:

Internet stvari (IoT) je dinamična, globalna, omrežna infrastruktura v kateri lahko teoretično poljubna stvar komunicira s katerokoli drugo stvarjo ali storitvijo. Realno pa so razvijalci aplikacij IoT soočeni z naborom omejitev in morajo pri razvoju aplikacij IoT sprejemati kompromise.
V magistrski nalogi sistematično preglejte obstoječe standarde in taksonomijo IoT. Formalizirajte nabor storitev, ki jih omogoča IoT. Analizirajte komunikacijske protokole, ki so na voljo v IoT. Preglejte obstoječe aplikacije in določite nabor aplikacijskih zahtev. Zgradite orodje za modeliranje aplikacij IoT, ki bo razvijalcu aplikacije IoT pomagal pri načrtovanju aplikacije IoT z izbranim obsegom omejitev. Izberite IoT aplikacijo in uporabite orodje za modeliranje aplikacij IoT za implementacijo prototipa.

Mentorica:

doc. dr. Mojca Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

*Hvala Tini, ker je še vedno izziv.*

*Hvala Janu za objeme in zakaje.*

*Hvala ati in hvala mami, ker nista nikoli preveč spraševala.*

*Hvala mentorici, doc. dr. Mojci Ciglarič, za odzivnost, pomoč, konstruktivne kritike in prilagajanje mojemu nenehno spreminjajočemu in bežečemu urniku.*

# Kazalo

# Kazalo slik

# Kazalo Tabel

# Seznam kratic

BLE..... nizkoenergijski Bluetooth (angl. *Bluetooth Low Evergy*)
EaaS..... vse oz. karkoli kot storitev (angl. *Everything as a Service*)
IoT ..... internet stvari (angl. *Internet of Things*)
IP..... internetni protokol (angl. *Internet Protocol*)
LED ..... svetleča diode (angl. *Light Emitting Diode*)
RFID ..... radijsko identifikacijo (angl. *Radio-frequency IDentification*)
Rx ..... sprejemanje podatkov (angl. *receive*)
TCP..... protokol za nadzor prenosa (angl. *Transfer Control Protocol*)
TCP/IP ..... sklad protokolov omrežja Internet
Tx..... pošiljanje podatkov (angl. *transmit*)

# Povzetek

V delu definiramo internet stvari kot skupino infrastruktur, ki omogočajo dostop, upravljanje in rudarjenje na podatkih, ki jih generirajo glede na okolico in interakcijo z uporabniki. Infrastrukture sestavljajo stvari, posredniki, podatkovne storitve, programski vmesniki, spletne in druge storitve, skupaj pa omogočajo prenos obdelavo in dostop podatkov s senzorjev ter nadzor aktuatorjev in spreminjanje notranjega stanja stvari. Stvari so naprave z določeno funkcijo, s senzorji za zaznavanje okolice in aktuatorji za manipulacijo okolice ter s sposobnostjo komunikacije z drugimi stvarmi, prehodi ali z drugimi omrežji.

Opremljeni z definicijo interneta stvari identificiramo nabor lastnosti, ki opisujejo aplikacije IoT in določimo gradnike, ki sestavljajo aplikacije IoT: stvari, posredniki ali prehodi ali stičišča, podatkovne shrambe in različne storitve. Določimo še osnovne arhitekture v katerih se lahko pojavljajo gradniki IoT. S tako določeno taksonomijo in pojmom aplikacije IoT zgradimo sistem za modeliranje aplikacij IoT, ki na podlagi zahtev aplikacije določi najprimernejšo arhitekturo in platformo, ki ju lahko izberemo za implementacijo aplikacije. S tem zmanjšamo prag za vstop v razvoj aplikacij IoT.

Orodje za modeliranje aplikacij IoT uporabimo za izdelavo pametne kuhinjske posode, ki se zaveda svoje teže in lahko posveti z diodo LED, v odvisnosti od notranjega stanja posode. Kot dokaz delovanja pripravimo vse komponente aplikacije IoT. Izdelamo tudi nadzorne in podporne spletne storitve, s katerimi se dotaknemo dogodkovno vodenega programiranja, komunikacije preko protokola *WebSocket* in težav s komunikacijami preko tehnologije *Bluetooth LE*.

V zaključku nakažemo možnost širitve taksonomije in skupnosti okrog koncepta testiranja modelov (angl. model *unit testing*). Nakažemo tudi možnost samodejnega odkrivanja lastnosti in priporočil s pomočjo strojnega učenja.

**Ključne besede:** internet stvari, iot, stvari, povezani sistemi, pametni sistemi, model, taksonomija, aplikacija, infrastruktura, arhitektura, komunikacije, websocket, posrednik, storitev.

# Abstract

We define the internet of things as a group of infrastructures for accessing, mining and management of data that is generated by the environment or by users. Infrastructures are comprised of things, transport elements, data services, programming interfaces, web services and other services. Infrastructures enable sensor data access, actuator control, and internal state change. Things are devices with a specific function and might have one or more sensors to sense the environment and zero or more actuators to act in the environment. Things can communicate with other things, gateways and other networks.

This definition of IoT allows us to identify a base set of attributes that can be used to describe IoT applications and basic components that appear in an IoT application such as things, gateways or hubs, data storages and services. We define the basic system architectures that are used in IoT applications. We build an IoT application model based on this taxonomy. The model takes a list of properties representing application requirements and proposes the best architecture and platform to use for building the desired IoT application. This reduces the friction to enter the world of IoT development.

We use the model to recommend the platform and an architecture for a smart kitchen container application. The smart kitchen container is aware of its weight and has an LED light that can be controlled through the network or through the internal state.

As a proof of concept we implement the IoT application including all the cloud based support services.

In conclusion we show a possible future direction of work developing a model unit testing tools. We also make suggestions for automatic feature learning and classification with machine learning.

**Keywords:** internet of things, iot, things, connected systems, smart systems, model, modeling, taxonomy, application, communication, websocket, transport, gateway, service.

# 1. Uvod

Internet stvari (angl. *Internet of Things*, IoT) lahko definiramo na najrazličnejše načine. Zdi se, da je definicij vsaj toliko kolikor je analitskih hiš. V svetu znanosti je nekoliko bolje, a tudi tu manjka sistematična definicija, ki bi pokrila vsa področja. In področij je res veliko. Pomislimo, govorimo o stvareh, ki so vsepovsod okrog nas in govorimo o Internetu, kjer nam je že pred časom zmanjkalo naslovnega prostora IPv4, torej $2^{32}$ oz. 4.294.967.296 naslovov. Ob takih številkah je napoved analitske hiše *Gartner Inc*, ki napoveduje v letu 2016 po svetu v uporabi 6.4 milijarde povezanih stvari [21], zelo verjetna. Sploh, če med stvari štejemo tudi naprave, ki so že danes del Interneta.

Internet stvari je veliko omrežje omrežij, kjer je vsaka stvar, ki postane del omrežja, potencialno povezljiva s katerokoli drugo stvarjo v tem omrežju. Stvari so torej del interneta stvari, če so povezljive, lahko neposredno v Internet, lahko v lokalna omrežja, ki imajo prehode v Internet, lahko pa so povezljiva s sebi enakimi in tvorijo zankasta (angl. *mesh*) omrežja. Zankasta omrežja, tudi omrežja vsak z vsakim (angl. *peer to peer*), so lahko samozadostna, lahko pa vsebujejo prehode preko katerih lahko komunicirajo z drugimi omrežji [11]. Vendar, če so zankasta omrežja samozadostna, potem ne morejo dostopati v druga omrežja in če smo napisali, da je internet stvari omrežje omrežij, potem samozadostna zankasta omrežja vsekakor niso del interneta stvari.

Naslednja težava smo tudi ljudje. Ljudje nismo stvari. Ampak, ljudje uporabljamo stvari. Nekatere definicije interneta svari človeka izločijo in poudarjajo, da gre v internetu stvari strogo za zaznavanje okolja, interakcijo z okoljem, komunikacijo med stvarmi, agregacijo podatkov in samodejnem odločanju o naslednjem stanju sistema. Tudi tovrstne definicije [10] sicer dopuščajo človeške posege na ravni poslovne analitike in konfiguracije sistema. Nekatere definicije definirajo internet ljudi, bodisi kot kontrast internetu svari, bodisi kot razširitev povezljivih stvari. Navsezadnje smo ljudje del fizičnega sveta v katerem so prisotne stvari. S stvarmi manipuliramo, jih uporabljamo in večinoma smo ljudje tisti, ki stvari konfiguriramo, vzpostavimo sisteme in podporno infrastrukturo ter jih nadziramo in varujemo.

Zdi se, da je pri definiciji stvari nekoliko manj zmede. Napisali smo že, da so stvari povezljive. Stvari imajo zmožnost zaznavanja okolja in možnost manipulacije z okoljem, kar omogoča

aktivno sodelovanje v poslovnih, informacijskih in družbenih procesih. Stvari lahko avtonomno sprejemajo odločitve ali pa prejmejo navodila za delovanje od storitev in drugih stvari. Stvari lahko nudijo storitve drugim stvarem in se povezujejo na storitve drugih stvari [12, 15].

Nekatere definicije so Internetu stvari zaupale tudi popolno povezanost – stvari in ljudje povezani kadarkoli, kjerkoli, s poljubnimi drugimi stvarmi in ljudmi, neodvisno od omrežne poti in storitev [14]. Vzemimo za primer hišo. Hiša ima zunaj senzor svetlobe in števec za porabo električne energije, v kuhinji senzor za dim in v hladilniku izdelke označene z nalepkami RFID (angl. *Radio-frequency identification*) in pametne posode, v dnevni sobi termostat in v garaži avto. V hiši sta vzpostavljena brezžično omrežje in priključek v Internet. Stanovalci imajo vsak svoj pametni telefon. Večina od naštetih stvari lahko postane del interneta stvari, če ji dodamo komponento za komunikacijo in procesiranje (termostat postane pametni termostat [12]), nekatere stvari (npr. pametni telefon) pa so že del interneta stvari. Teoretično lahko sedaj vse stvari v hiši komunicirajo druga z drugo kakor tudi s stvarmi in storitvami v Internetu. Praktično pa hitro naletimo na težave zaradi različnosti naprav. Števec za porabo električne energije ima praktično neomejeno zalogo le-te, pametna posoda v hladilniku pa je obsojena na baterijo velikosti kovanca. Pametni telefon mora venomer preverjati in jasno prikazati, če smo prejeli elektronsko pošto, na senzor dima pa lahko pozabimo vse do nezgode ali požara v kuhinji. Nalepke RFID lahko uporabimo za identifikacijo hrane v hladilniku, vendar pa jih z večino pametnih telefonov ne bomo mogli prebrati.

Upamo, da smo na tem mestu vsaj malo izzvali pogled bralca na internet stvari in ga prepričali, da potrebujemo natančneje opredeljene pojme ali pa nabor omejitev, ki veljajo v danem kontekstu in nam tako poenostavijo proučevanje.

Postavimo se za trenutek v vlogo inženirja računalništva, ki želi razviti pametno posodo, ki bo spremljala svojo težo in ob določenih dogodkih prižgala diodo LED (angl. *Light-Emitting Diode*). Nadobudni inženir bo najprej naletel na zmedeno terminologijo, v drugi fazi pa še na poplavo različnih platform, ki so tako ali drugače povezane z internetom stvari. Kje naj začne? Katero platformo naj izbere? Kakšne podporne storitve potrebuje? V kakšni arhitekturi in v kakšnih soodvisnostih bodo nastopale stvari in podporne storitve? Vsekakor je tak svet mnogo lepši kot svet, kjer tehnologija sploh ni na voljo ali pa je na voljo samo ena, za katero smo lahko vnaprej prepričani, da našega problema ne bo v celoti rešila.

V tem delu bomo pregledali področje in skušali podati definicijo interneta stvari, ki bo smiselna v kontekstu razširjenega pametnega doma/hiše. Razlikovalne lastnosti stvari bomo skušali organizirati v taksonomijo in v nabor pravil, ki bo na eni strani razširljiv s prihajajočimi tehnologijami na drugi strani pa bo znal modelirati aplikacije v današnjem internetu stvari. Izdelali bomo model, v katerega bo lahko naš nadobudni inženir vnesel zahteve, model pa bo zanj izbral najprimernejšo arhitekturo in strojno platformo za izvedbo.

V drugem poglavju bomo najprej pregledali definicije interneta stvari in obstoječe poskuse taksonomij interneta stvari. Potem bomo identificirali sistemske arhitekture, ki lahko nastopijo v internetu stvari. Na koncu drugega poglavja bomo definirali internet stvari za kontekst tega dela. Definirali bomo tudi pojem aplikacije interneta stvari.

V tretjem poglavju se bomo lotili modeliranja aplikacij interneta stvari. Najprej bomo identificirali nabor lastnosti, ki jih imajo lahko stvari. Potem bomo razdelali postopek modeliranja aplikacije interneta stvari in prikazali primer modela.

V četrtem poglavju bomo uporabili predhodno razvit model za pomoč našemu mlademu inženirju. Identificirali bomo nabor zahtev, pripravili model in glede na dobljen rezultat razvili aplikacijo za internet stvari.

V petem poglavju bomo podali sklepne ugotovite, smernice za nadaljnje delo in vizijo, kje in kako se bi lahko naš model v prihodnosti uporabljal.

# 2. Internet stvari

V literaturi najdemo različne definicije pojma internet stvari. Definicije se razlikujejo po obsegu, po vključenosti stvari in po taksonomiji, zato bomo v nadaljevanju najprej pregledali literaturo in povzeli druge definicije, nato pa določili svojo definicijo, ki jo bomo uporabljali v preostanku tega dela.

## 2.1. Pregled literature

Besedna zveza internet stvari se prvič pojavi leta 1999 v predstavitvi za podjetje *P&G* [3]. Avtor je želel z besedno zvezo poudariti, da so podatki v današnjih informacijskih sistemih večinoma odvisni od ljudi in da moramo v prihodnosti razviti informacijske sisteme, ki se bodo zavedali svoje okolice. Za primer informacijskega sistema odvisnega od ljudi vzemimo informacijski sistem dostavne službe, kjer ima vsak paket črtno kodo, ki jo uslužbenci odčitajo, ko paket potuje skozi logistični center. Podatki v informacijskem sistemu dostavne službe so odvisni od dejanj uslužbencev, ne pa od paketov (stvari) samih. hongkonško letališče se je podobnega problema lotilo na drugačen način. Tam vsak kovček dobi svojo oznako RFID. Ko kovček potuje skozi letališče, ga antene RFID samodejno zaznajo in ustrezno posodobijo podatke v informacijskem sistemu. Podatki o kovčku tako vsaj na poti skozi letališče niso več odvisni od ljudi [16, 13].

*McKinsey Global Institute* definira v svojem poročilu o stanju IoT [10] internet stvari kot porazdeljeno omrežje senzorjev in sprožil (angl. *actuators*) povezanih z računalniškimi sistemi. Stvari, ki zahtevajo vhodne podatke od ljudi, niso del interneta stvari. Tudi pri tej definiciji je pri zajemu in zbiranju podatkov poudarjena samodejnost in avtonomnost komponent. Seveda pa to ne pomeni, da ljudje niso vključeni v internet stvari. Nasprotno, analitiko, obdelavo in uporabo zajetih podatkov si težko predstavljamo brez interakcije z ljudmi. Če se strogo držimo definicije, potem je senzor srčnega utripa, ki ves čas meri naš utrip in ga skupaj s časom meritve pošlje storitvi v računalniškem oblaku, del interneta stvari. Na drugi strani pa aplikacija, ki jo uporabnik požene, pritisne na gumb za meritev utripa in utrip skupaj s časom meritve požene storitvi v računalniškem oblaku, ni del interneta stvari.

V članku *A Simple Explanation of The Internet of Things*, pomeni internet stvari povezovanje katerekoli naprave s stikalom za vključitev in izključitev bodisi v Internet bodisi s katerokoli drugo tako napravo [26]. Ta definicija je širša od predhodnih, saj vključuje na primer tudi: pametne telefone, pečice, luči, nosljivo tehnologijo (angl. *wearables*) in računalnike. Če smo čisto striktni pa ne vključuje kovčkov s hongkonškega letališča, saj niti kovčki niti oznake RFID nimajo stikal za vklop in izklop.

*Gartnerjev* slovar za informacijsko tehnologijo definira internet stvari kot omrežje fizičnih objektov, ki vsebujejo tehnologijo za komunikacijo in za zaznavanje ali interakcijo z njihovim stanjem ali z njihovo okolico [20]. Ta definicija je podobno široka kot prejšnja, hkrati pa se izogne problemu s hongkonškimi kovčki.

Avtorji *IDC-jevega* globalnega poročila o internetu stvari [9] so definirali internet stvari kot omrežje omrežij, ki ga sestavljajo enolično določljive končne točke oz. stvari. Stvari imajo IP povezljivost in komunicirajo globalno ali lokalno brez posredovanja ljudi.

Ameriška državna fundacija za znanost (angl. *National Science Foundation*) je v svojih programih nejasno definicijo interneta stvari nadomestila z bolj neposrednim izrazom – fizične povezane naprave. Ameriški državni inštitut za standarde in tehnologijo (angl. *National Institue of Standards and Technology*) je podobno nadomestil pojem internet stvari s pojmom Kiber-fizični sistemi (angl. *Cyber-physical system*), ki ga definirajo kot pametni sistem, ki vključuje načrtovano interakcijo omrežij fizičnih in računalniških komponent [2].

## 2.1.1. Pregled taksonomij interneta stvari

Da bomo lahko določili skupne lastnosti in gradnike v internetu stvari, bomo v nadaljevanju pregledali kako so drugi opredelili taksonomijo interneta stvari na različnih področjih.

### 2.1.1.1. Delitev interneta stvari – *IDTechEx*

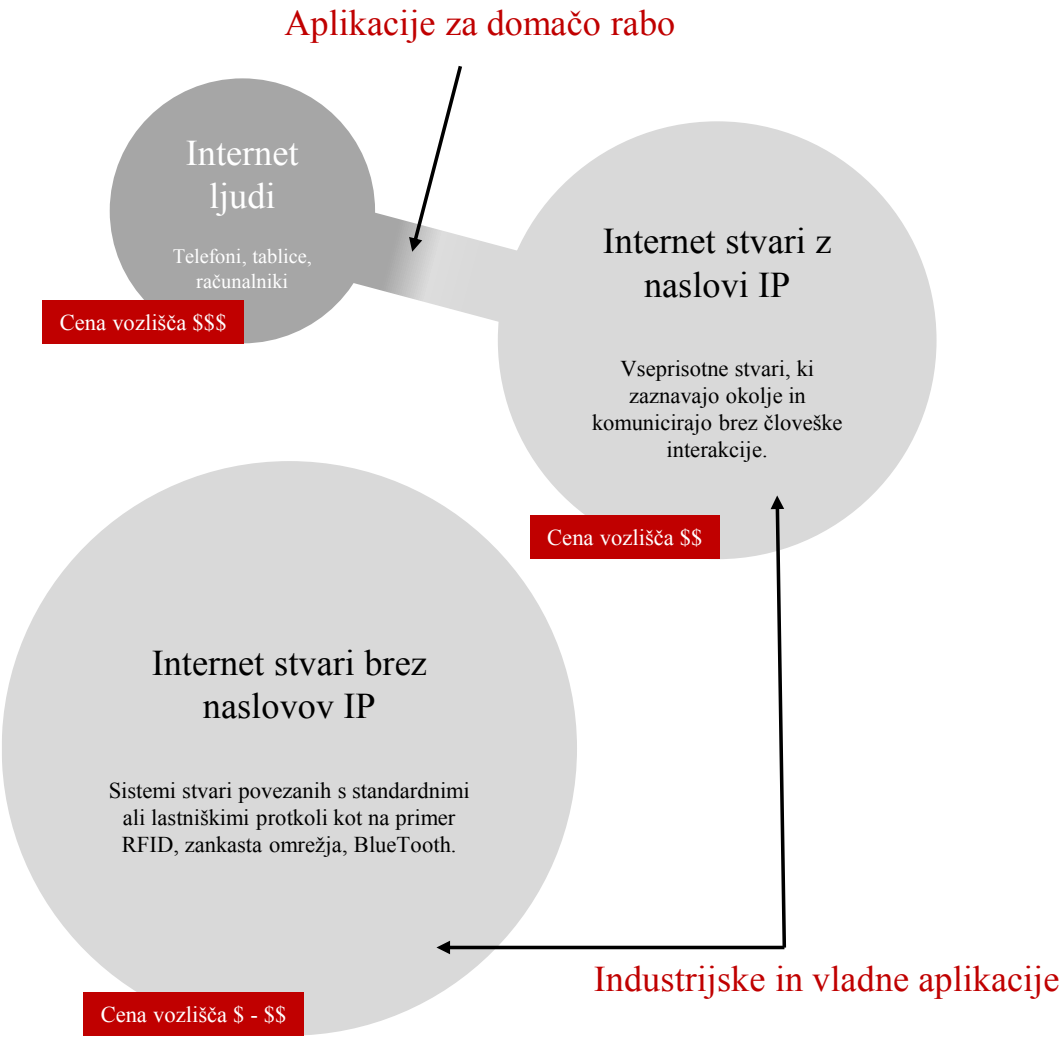
*IDTechEx* je organizacija, ki pripravlja neodvisne raziskave trga, poslovno analitiko in organizira dogodke o prihajajočih tehnologijah. Internet stvari delijo na (Slika 1):

- Internet ljudi: sestavljajo ga v Internet povezane naprave kot so pametni telefoni, tablični računalniki, prenosni in osebni računalniki.
- Internet stvari z naslovi IP: vseprisotne pametne stvari, ki zaznavajo svoje okolje, so povezane v Internet in komunicirajo in posodabljajo svoje notranje stanje brez človeškega posredovanja.
- Internet stvari brez naslovov IP: sistemi stvari, senzorjev in identifikatorjev, ki uporabljajo lastniške ali standardizirane protokole za komunikacijo. Internet stvari brez

naslovov IP predstavljajo na primer sistemi RFID, globalni lokacijski sistemi, senzorji povezani v zankasta omrežja (angl. mesh network).

Aplikacije IoT so razdeljene v dva segmenta:

1. Aplikacije za domačo rabo ali potrošniške aplikacije: nosljiva tehnologija, avtomatizacija doma, skrb za zdravje, telesna pripravljenost, dobro počutje in pomoč na domu, potrošniške storitve in zabava, vozila.
2. Industrijske in vladne aplikacije: nadzor infrastrukture, pametna mesta, javna razsvetljava, javni prevoz, nadzor porabe energije, pametno električno omrežje (angl. *smart grid*), procesna avtomatika, varnost, kmetijstvo.




*Slika 1. Taksonomija IoT IdTechEX*

Osnovni gradniki interneta stvari so:

- senzorji in integrirana vezja,
- naprave,
- prehodi,
- programska oprema.

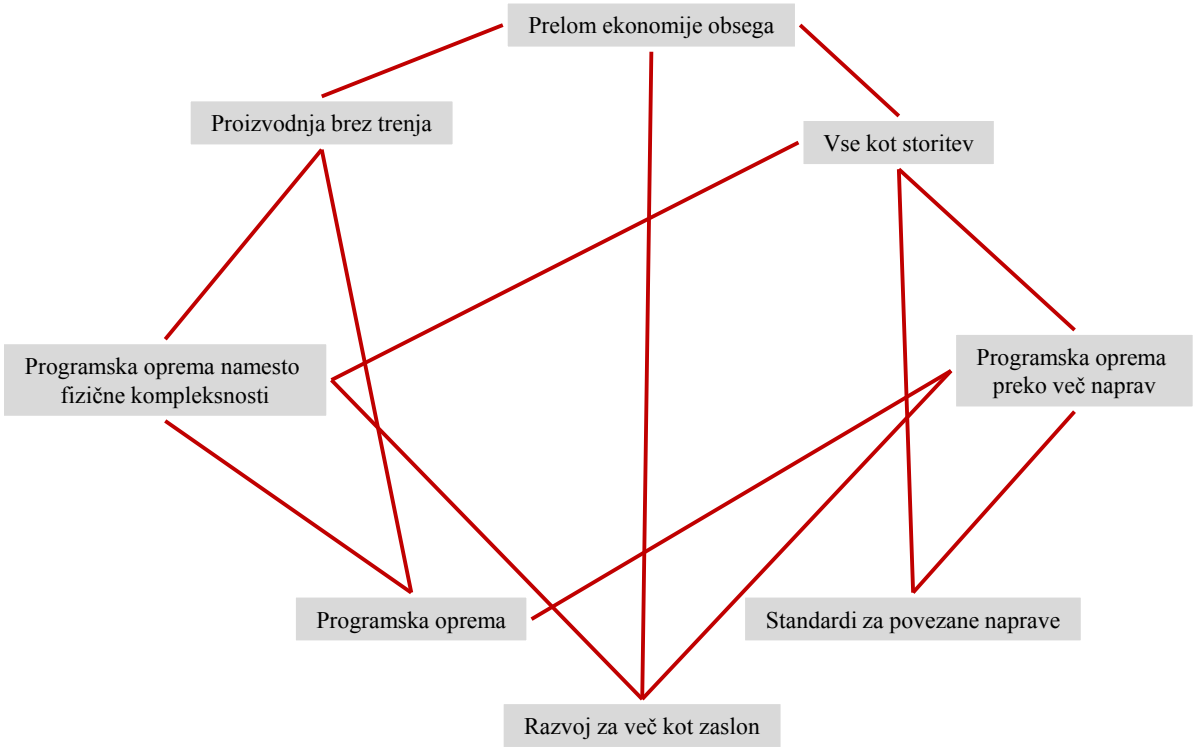
Iz osnovnih gradnikov in naštetih aplikacij IoT lahko sklepamo, da je aplikacija IoT določena kot nabor senzorjev, naprav, prehodov in programske opreme, ki sodelujejo pri podpori določenega poslovnega procesa ali rešujejo določen problem [23, 7].

#### 2.1.1.2. Osnovni koncepti – *O'Reilly*

V knjigi *Kaj je internet stvari* se avtorja izogneta definiciji interneta stvari, namesto definicije opredelita osem ključnih konceptov, ki so del vsakega modernega, zanimivega projekta, ki vključuje stvari, kot smo jih določili na začetku poglavja. Koncepti veljajo za projekte od dodatkov za pametne telefone do motorjev za reaktivna letala [8]. Slika 2 prikazuje koncepte in njihovo medsebojno prepletanje.

Osnovni koncepti interneta stvari so:

1. Prelom ekonomije obsega (angl. *disrupting economies of scale*): gre za zmanjševanje upora pri vstopu izdelka na trg. Tako kot *Amazon Web Services* (in konkurenca) omogočajo hiter razvoj in objavo aplikacij, omogoča internet stvari hitro prototipiranje naprav in storitev. Platforme za množično financiranje (angl. *crowdfunding*), kot sta *Kickstarter* in *IndieGoGo* pa omogočajo učinkovito zbiranje zagonskega kapitala v kombinaciji z raziskovanjem trga.
2. Programska oprema: vsa podjetja postajajo podjetja za razvoj programske opreme. Tradicionalna proizvodnja podjetja, npr. *Ford*, *GE*, razvijajo vse več programske opreme. Podjetja znajo materiale spremeniti v izdelke, programska oprema pa lahko izdelke poveže v internet stvari. Vzemimo za primer zabojnik za odpadke, ki je zgolj kup plastike. Zabojniku lahko dodamo programsko opremo, ki nas bo opozorila, da je zabojnik poln, naročila odvoz ali opozorila smetarje, da so ga pozabili izprazniti.
3. Proizvodnja brez trenja (angl. *frictionless manufacturing*): proizvodne kapacitete so lažje dostopne kot kdaj koli prej in na voljo so nam podjetja, ki poenostavijo oskrbovalno verigo (angl. *supply chain*) na uporabo aplikacije, pošiljanje naročila in sprejem izdelkov [27]. Hkrati so 3D tiskalniki in 3D bralniki (angl. *scanner*) poenostavili proizvodne procese in zmanjšali čas od ideje do prototipa.




*Slika 2. Osnovni koncepti IoT - O'Reilly.*

4. Programska oprema preko več naprav: prava inteligenca izhaja iz povezanosti več naprav. Tako postane električni avto pametnejši, če se začne polniti takrat, ko mu električno omrežje sporoči, da je cena energije nižja zaradi pomanjkanja povpraševanja. Tudi električno omrežje je lahko pametno in se o ceni energije odloča na podlagi dejanske trenutne porabe. Takšno povezljivost med sistemi dosežemo z gradnjo odprtih in standardiziranih programskih vmesnikov (angl. *application programming interface,* API).
5. Standardi za povezane naprave: razlog, da se lahko v Internet povežemo iz tako različnih naprav so odprti standardi. Žal internet stvari danes pesti pomanjkanje odprtih standardov, a vendar se stvari izboljšujejo. *Philips* je odprl in standardiziral programski vmesnik za upravljanje pametnih luči, *Thing System* določa nabor programskih vmesnikov za nestandardne naprave [28], *ZigBee* pa postaja standard za avtomatizacijo doma.
6. Razvoj za več kot zaslon: aplikacije v internetu stvari večinoma nimajo standardnih uporabniških vmesnikov kot sta zaslon in tipkovnica. Pas za spremljanje telesne aktivnosti *Fitbit Flex* nima zaslona, namesto tega sporoča svoje stanje z nekaj svetlečimi diodami in vibracijami. Podoben primer je tudi aplikacija za navigacijo na kolesu *BeeLine*, kjer ima zaslon le zmožnost prikazovanja puščic levo in desno. Vse več je tudi vmesnikov, ki prepoznajo glasovne ukaze, npr. *Amazon Echo*.

7. Vse kot storitev (angl. *Everything as a Service*, EaaS): ta koncept pravi, da naj podjetja delajo tisto, v čemer so najboljša in to izpostavijo kot storitev. Agencija za izposojo avtomobilov (angl. *rent-a-car*) bi tako lahko izpostavila svoj programski vmesnik, ki ga lahko povežemo s spletno trgovino in zemljevidom. Kombinacija aplikacij se lahko sedaj v odvisnosti od posameznikovih nakupovalnih želja odloči za javni prevoz, avto, kombi ali tovornjak in opravi ustrezne rezervacije.
8. Programska oprema namesto fizične kompleksnosti: včasih lahko kompleksno nadgradnjo fizične stvari nadomestimo s programsko opremo, ki omogoči učinkovitejše delovanje. Za primer vzemimo domačo peč. Nadgradnja ali zamenjava peči je lahko zelo draga, vendar pa lahko njeno učinkovitost vsaj do neke mere izboljšamo s pametnim termostatom.

#### 2.1.1.3. Skupne lastnosti IoT – *Internet2*

*Internet2* je akademski ponudnik povezav v Internet v ZDA. V okviru organizacije delujejo delovne skupine, kjer lahko akademske in raziskovalne inštitucije odprto rešujejo zapletene tehnološke in raziskovalne probleme. Ena od delovnih skupin se ukvarja z internetom stvari.

Skupina ugotavlja, da je taksonomija IoT pomembna in potrebna za standardizacijo in razvoj interneta stvari v prihodnosti, vendar pa zaradi trenutne dinamičnosti področja ne poda svojih priporočil [2].

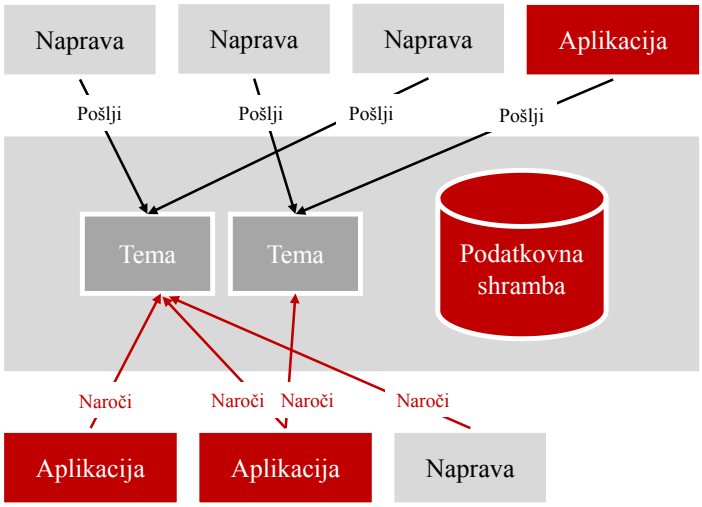
Skupina je identificirala skupni nabor lastnosti IoT:

- Povezljivost: večina fizičnih stvari je povezljiva zaradi omejenega prostora za hranjenje podatkov in posledično potrebe po nenehnem prenosu podatkov. Povezljivost stvari je lahko trajna ali v presledkih (angl. *intermittent*). Stvari se povezujejo s centralno točko, ki agregira podatke in jih posreduje naprej v podatkovne shrambe za podatkovno analizo.
- Osredotočenost na podatke: model IoT vključuje zbiranje, prenos in predstavitev podatkov. Stvari zaznavajo in zbirajo podatke iz okolice, na primer telesno temperaturo, nivo sladkorja v krvi, temperaturo okolice, relativno vlažnost. Stvari v IoT prenašajo tudi kontrolne podatke, ki lahko aktivirajo procese, prižigajo in ugašajo stvari, premikajo motorje, ipd.
- Nizka poraba: večina stvari v IoT, je prenosnih, porazdeljenih, premakljivih ali zunanjih, kar pomeni, da črpajo električno energije iz šibkih virov, kot so baterije, sončne celice ali PoE (angl. *Power over Ethernet*).
- Minimalni viri: večina stvari v IoT je zelo omejena z viri. Naprave so majhne velikosti, uporabljajo procesorje z nizko porabo energije, imajo omejen prostor za shranjevanje podatkov in počasno podatkovno povezavo.

- Uporaba storitev v oblaku: storitve računalništva v oblaku so zelo primerne za porazdeljeno naravo naprav IoT in njihovo potrebo po centraliziranem zbiranju podatkov. Večina ponudnikov infrastrukture v oblaku že ponuja rešitve IoT [17, 24, 22].

Skupina je definirala tudi lastnosti modela za izmenjavo podatkov:

- Model pošiljatelj/naročnik: podatkovni prenosi med napravami so večinoma asinhroni in uporabljajo model pošlji in naroči. V tem modelu pošiljajo končne točke podatke v vnaprej določene teme (angl. *topic*) na podatkovni shrambi. Druge naprave se naročijo na isto temo, če želijo podatke prejeti. Posamezna točka je lahko tako pošiljatelj kot naročnik za več tem (Slika 3).
- Lahek protokol za sporočanje: zaradi omejenih virov v IoT morajo biti protokoli za izmenjavo sporočil zanesljivi, lahki in z nizko latenco. Trenutno je priporočena uporaba protokolov HTTP/HTTPS in MQTT. Slednji podpira model pošiljatelj naročnik [25].
- Podatkovna shramba: zaznani podatki so nestrukturirani, kar pomeni, da so za njihovo shranjevanje najprimernejše podatkovne baze NOSQL.




*Slika 3. Model za izmenjavo podatkov v IoT.*

### 2.1.1.4. Taksonomija IoT za nadzor osebne lastnine

Avtorji članka dodajo v internet stvari predmete, ki jih zazna na primer kamera v pametnem telefonu. Taki predmeti tako postanejo sledljivi in vključeni v internet stvari, kljub temu, da nimajo svojih procesorskih zmogljivosti.

V članku so identificirane naslednje osnovne komponente IoT:

- Pametne naprave: naprave ali mehanizmi, ki lahko strežnikom ali drugim pametnim napravam sporočijo svojo prisotnost. Pametne naprave lahko vsebujejo senzorje za zajem podatkov iz okolice in znajo tudi te sporočiti strežnikom ali drugim pametnim napravam.
- Trajna vozlišča (angl. *persisent node*): so del omrežja s sposobnostjo srednje ali dolgoročne hrambe podatkov.
- Zbirke: stvari, ki imajo skupne lastnosti, posamezna stvar iz zbirke pa nima enoličnega identifikatorja. Stvari lahko postanejo enolično določene šele, če jih povežemo z drugimi podatki iz IoT. Zbirko si lahko predstavljamo kot izdelek z določeno črtno kodo. Črtna koda določa tip izdelka, izdelka pa ne identificira enolično.
- Akterji/čuteči agenti (angl. *sentient agent*): stvari, ki imajo svobodno voljo in lahko sprejemajo odločitve o svojih dejanjih in akcijah. Tipičen predstavnik te komponente je človek.
- Delno avtonomni agenti: agent s sposobnostjo avtonomnega izvajanja akcij, ki jih je predhodno določil skrbnik sistema.
- Pokvarljive dobrine: zbirke, ki imajo kratko življenjsko dobo, npr. sadje in zelenjava.
- Prehodi: povezava med različnimi omrežji. Prehod je v tem kontekstu mišljen na višjem nivoju od prehoda v računalniških omrežjih – blagajna je na primer prehod med človekom (kupcem) in banko, ki omogoča prenos denarja.

Članek govori tudi o možnostih za povezovanje v IoT. Ena možnost je centralizirano povezovanje s prehodom (zvezda). Druga možnost pa so omrežja enak z enakim oz. zankasta omrežja [4].

#### 2.1.1.5. Analiza trga IoT – *IDC*

Avtorji *IDC-jeve* analize trga IoT [9] identificirajo naslednje osnovne gradnike interneta stvari:

- Inteligentni in razširjeni tradicionalni vgradni sistemi.
- Povezljivost in storitve: stvari imajo IP povezljivost in lahko izpostavijo storitve.
- Platforme: naprava, omrežje in omogočanje aplikacij.
- Analitika in družbeno poslovanje.
- Aplikacije.
- Varnost.
- Strokovne storitve.

V analizi so podani tudi prihodnji izzivi na področju interneta stvari:

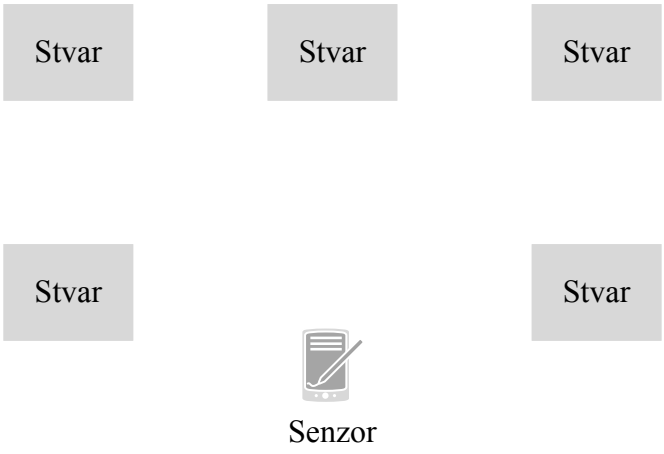
- Pomanjkanje formalnih standardov.
- Globalna skalabilnost je vprašljiva zaradi pomanjkanja povezljivosti v nekaterih delih sveta, kjer ni niti GSM niti satelitske pokritosti.
- Dinamičen in razvijajoč ekosistem za razvoj aplikacij ponuja nešteto možnosti za razvoj stvari in aplikacij, vendar pa to pomeni povečano tveganje in nenehno prilagajanje spreminjajočim se platformam.
- Zasebnost: premalo formalnih standardov za zagotavljanje zasebnosti v IoT, kjer so stvari enolično določljive.

## 2.2. Arhitektura

Preden definiramo pojem internet stvari kot ga bomo uporabljali v preostanku dela, poglejmo na kakšen način so lahko stvari v internetu stvari povezane med seboj in z drugimi omrežji.

### 2.2.1. Nepovezana arhitektura

Stvari v nepovezani arhitekturi (Slika 4) so samostojne in ne komunicirajo med sabo. Do komunikacije in interakcije pride samo ob prisotnosti senzorja ali zbiratelja.



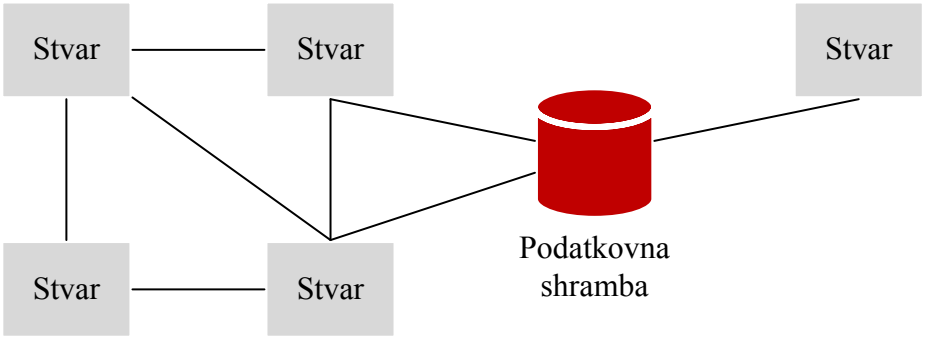

*Slika 4. Nepovezana arhitektura.*

Za primer lahko vzamemo notranjo navigacijo po prostoru. V prostoru so statično nameščeni oddajniki BLE (angl. *Bluetooth Low Energy*). Naš telefon (senzor) jih zazna, ko se jim približamo in na podlagi tega določi svojo lokacijo.

### 2.2.2. Samozadostna arhitektura

Stvari v samozadostni arhitekturi (Slika 5) komunicirajo med seboj in so povezane v zankasto omrežje. Stvari nimajo nujno neposredne povezave druga z drugo, imajo pa sposobnost

usmerjanja podatkov. Tako lahko vsaka stvar komunicira s katerokoli drugo stvarjo. V arhitekturi je lahko, kot stvar s posebno vlogo, prisotna lokalna shramba podatkov.



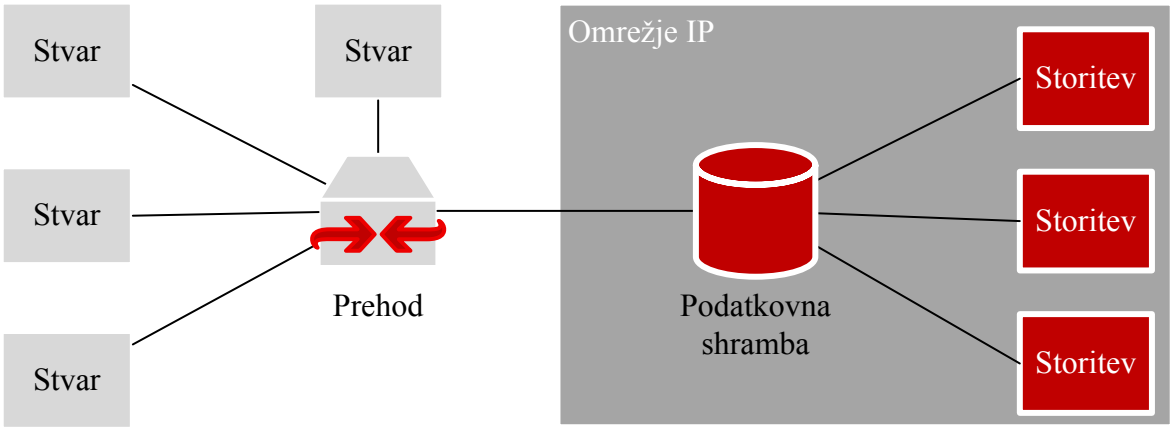

*Slika 5. Samozadostna arhitektura.*

Seveda se lahko na tem mestu upravičeno vprašamo, če samozadostna in nepovezana arhitektura sploh sodita v internet stvari, saj obema manjka povezava v Internet. Menimo, da ju je zavoljo popolnosti smiselno vključiti. Še več, glede na varnostne pomanjkljivosti nekaterih današnjih aplikacij IoT menimo, da bi bilo bolje, da bi uporabile samozadostno arhitekturo.

### 2.2.3. Posredniška arhitektura

V posredniški arhitekturi (Slika 6) se vse stvari povezujejo na centralno napravo oz. stvar, ki igra vlogo posrednika. Posrednik je lokalno povezan s stvarmi, na drugi strani pa globalno s spletnimi storitvami, oddaljeno shrambo in drugimi storitvami v drugih omrežjih.

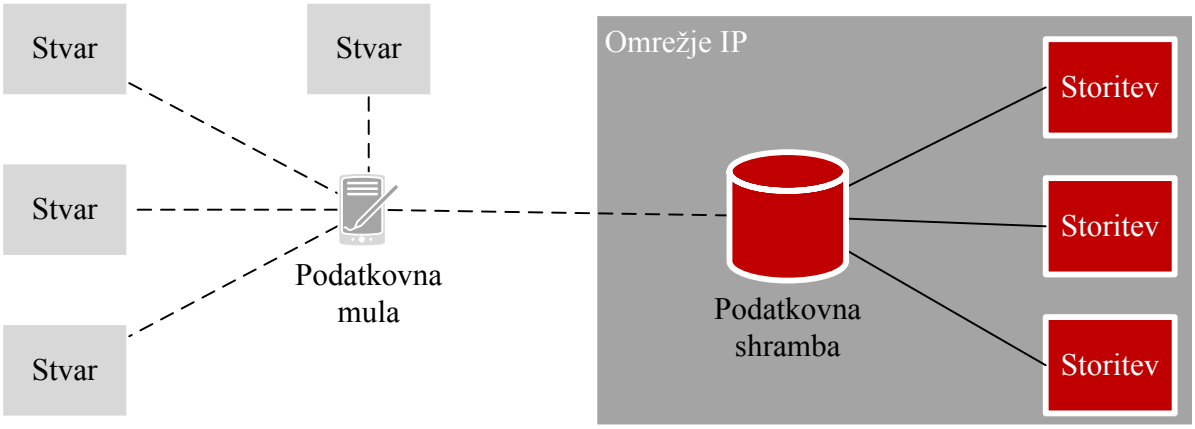
Arhitekturo srečamo v primerih, kjer so stvari lokalno povezljive, nimajo pa sklada IP in ne morejo komunicirati neposredno s storitvami izven lokalnega omrežja. Zamislimo si lahko pametno hišo, kjer so stvari lokalno povezane v omrežje ZigBee, globalno povezljivost in dostop do stvari v hiši iz poljubne lokacije pa omogoča stičišče ali prehod.




*Slika 6. Posredniška arhitektura.*

### 2.2.4. Občasno povezana arhitektura

Občasno povezano arhitekturo (Slika 7) imamo takrat, ko stvari samostojno zaznavajo in morda tudi delno agregirajo podatke, nimajo pa povezljivosti s podatkovno shrambo ali drugimi storitvami v omrežju Internet. Za prenos podatkov se zanašajo na podatkovne mule (angl. data mule), ki pa jih lahko stvari za prenos podatkov uporabijo le začasno in v nepredvidljivih časovnih intervalih.

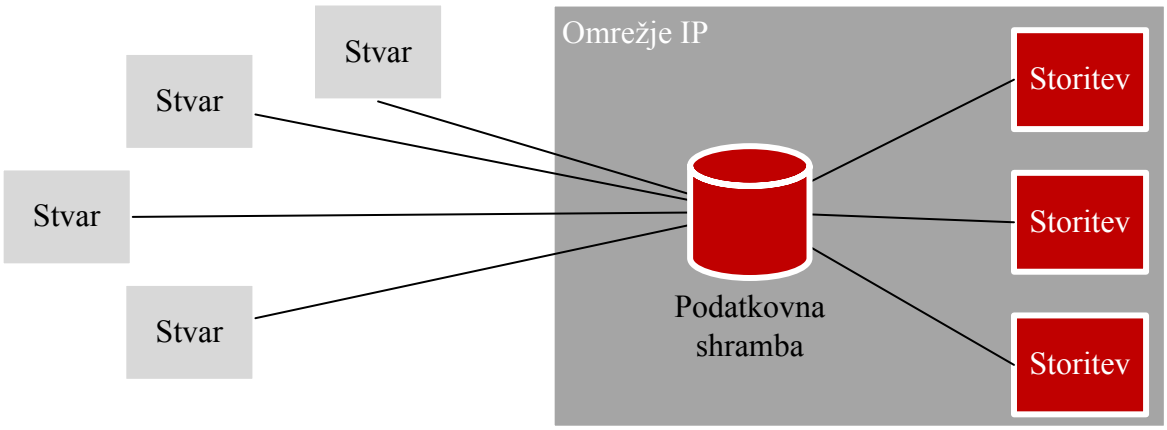



*Slika 7. Občasno povezana arhitektura.*

Recimo, da smo razvili stvar s senzorjem za štetje prometa. Stvar lahko popolnoma avtonomno šteje promet, zaradi nedostopnosti omrežja pa ni povezana v Internet. Za prenos podatkov lahko izkoristimo mimoidoče pametne telefone, ki sicer na lokaciji stvari prav tako niso povezane, bodo pa povezane kasneje, ko bo na voljo omrežje. Telefon tako postane podatkovna mula, stvar s senzorjem pa del občasno povezane arhitekture. Tehničnim podrobnostim in vprašanju zasebnosti smo se na tem mestu namenoma izognili.

### 2.2.5. Neposredno povezana arhitektura

V neposredno povezani arhitekturi (Slika 8) imajo vse stvari ves čas na voljo povezljivost IP. S tako arhitekturo se komunikacija med stvarmi, podatkovno shrambo in drugimi storitvami precej poenostavi. Seveda pa takšna arhitektura ni vedno izvedljiva bodisi zaradi omejitev porabe električne energije bodisi zaradi odsotnosti komunikacijskega omrežja.



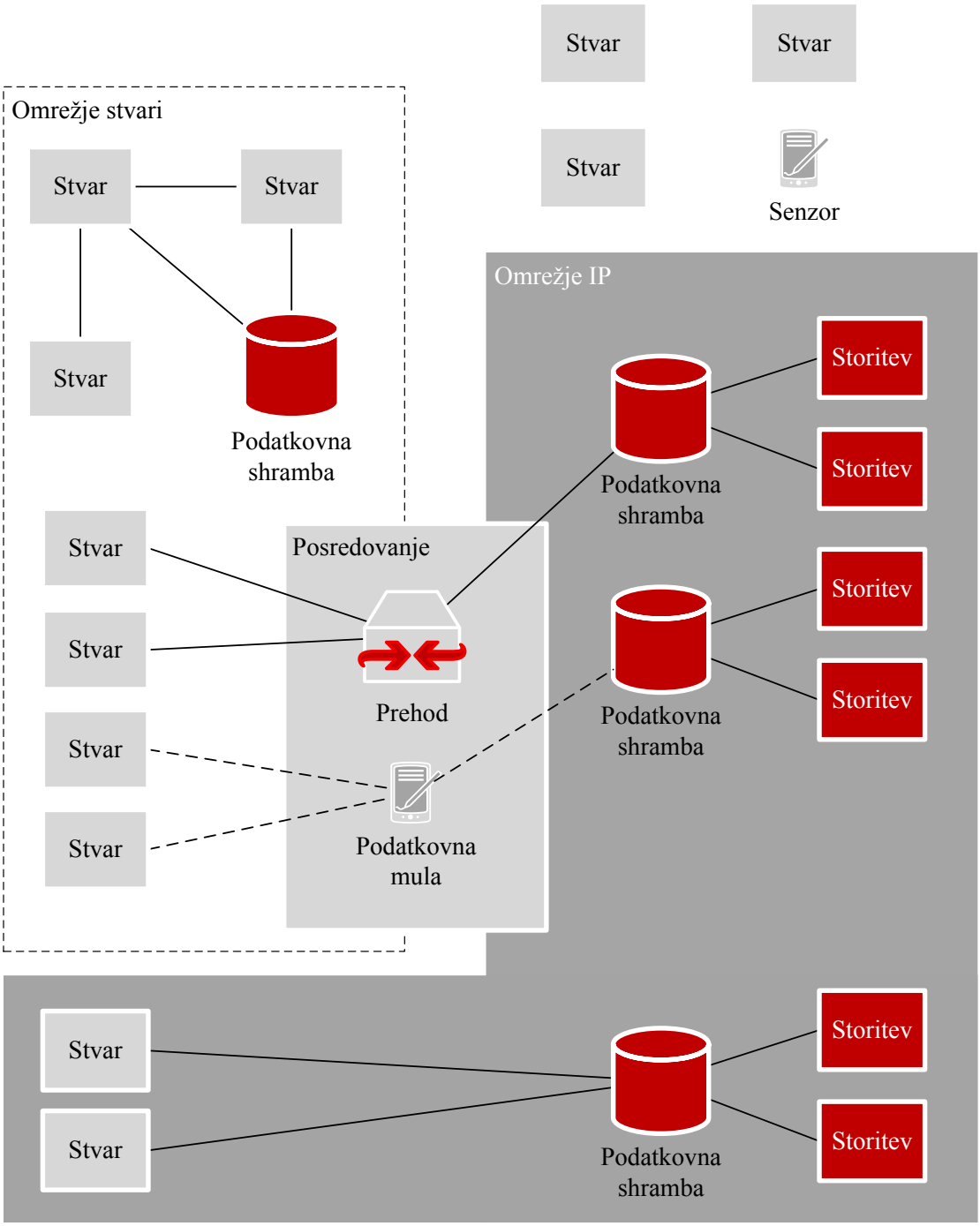

*Slika 8. Neposredno povezana arhitektura.*

### 2.2.6. Splošna arhitektura

Zavoljo boljšega pregleda smo v splošno arhitekturo združili vse podane arhitekture (**Napaka! Vira sklicevanja ni bilo mogoče najti.**). Opazimo lahko, da je del arhitekture v omrežju IP enak ne glede na arhitekturo. Spreminja pa se omrežje stvari in način posredovanja podatkov med omrežjem stvari in omrežjem IP [6].

Ključne komponente arhitekture so:

- Omrežje stvari: povezuje stvari s prehodi, stičišči ali podatkovnimi mulami. Povezave so v omrežju stvari lahko ožičene, npr. *Ethernet*, lahko pa so brezžične, npr. BLE, *Wi-Fi*, *ZigBee*. Podatkovne mule so lahko pametni telefoni, tablice, majhni vgradni računalniki in druge stvari.
- Posredovanje: prenos podatkov med omrežjem stvari in omrežjem IP, kjer se nahajajo podatkovna shramba, storitve in morebitni nadzorni strežniki.
- Podatkovna shramba: podatkovna shramba je lahko centralizirana ali porazdeljena. Stvarem mora nuditi možnost shranjevanja in branja podatkov. Dostop do podatkov mora omogočati tudi drugim storitvam v omrežju IP.
- Storitve: spletne storitve za dostop in manipulacijo podatkov; spletne storitve za nadzor stvari in morebitnih aktuatorjev; storitve za poslovno analitiko in podatkovno rudarjenje; storitve za uporabniške vmesnike; storitve za komunikacijo med stvarmi; ipd.




*Slika 9. Splošna arhitektura.*

Tako podatkovna shramba kot storitve so lahko implementirane centralizirano ali porazdeljeno. Storitve lahko tečejo v računalniškem oblaku ali več različnih računalniških oblakih, lahko tečejo v podatkovnem centru podjetja ali v hibridni kombinaciji obeh. Podrobnosti implementacije podatkovne shrambe in storitev nas v tem delu ne zanimajo, zato predpostavimo, da lahko do njih dostopamo, če imamo povezljivost IP.

Omenili smo storitve za uporabniške vmesnike, nikjer v arhitekturi pa nismo prikazali uporabnikov. Kljub temu, da menimo, da smo ljudje del interneta stvari, saj stvari nadziramo, manipuliramo in uporabljamo, kakor tudi, da stvari ponujajo zanimive možnosti za drugačno interakcijo in za uporabo nestandardnih uporabniških vmesnikov, bomo tudi za ta del privzeli, da ga lahko rešimo s spletno storitvijo, v kontekstu stvari pa nas bo zanimalo samo ali je možna direktna interakcija med uporabnikom in stvarjo.

## 2.3. Definicija IoT

Za definicijo interneta stvari se bomo v veliki meri oprli na [6] in [10]. V nadaljevanju dela bomo izraz internet stvari in kratico IoT razumeli kakor jih definiramo v naslednjih odstavkih.

**Stvar** je naprava z določeno funkcijo. Lahko ima senzorje za zaznavanje okolice. Lahko ima aktuatorje za manipulacijo z okolico. Stvar ima lahko sposobnost komunikacije z drugimi stvarmi, s prehodi ali z drugimi omrežji (npr. Internet).

Stvari so del **infrastrukture** in skupaj s posredniki, podatkovnimi storitvami, programskimi vmesniki, spletnimi in drugimi storitvami omogočajo:

- prenos, obdelavo in dostop do podatkov, ki prihajajo iz senzorjev in od uporabnikov;
- nadzor aktuatorjev in spreminjanje notranjega stanja stvari na podlagi uporabnikove akcije ali stanja drugih stvari.

**Internet stvari** je skupina infrastruktur, ki omogočajo dostop, upravljanje in rudarjenje na podatkih, ki jih generirajo glede na okolico in interakcijo z uporabniki.

## 2.4. Aplikacija IoT

S pojmom aplikacija interneta stvari oz. **aplikacija IoT** bomo označili nabor stvari, podatkovnih shramb, storitev in posrednikov povezanih v eno od arhitektur definiranih v 2.2, z namenom reševanja vnaprej določene množice primerov uporabe.

Zamislimo si moderno, pametno kuhinjo, v kateri želimo skuhati italijansko rižoto po receptu, ki smo ga našli na svetovnem spletu. Če za primer uporabe vzamemo tehtanje sestavin, potem je lahko naša aplikacija IoT precej preprosta in sestavljena iz pametne tehtnice (stvari) in aplikacije na našem pametnem telefonu, ki bo znala prikazati trenutno težo. Če primer uporabe razširimo na proces priprave hrane, bi naša aplikacija IoT lahko obsegala naslednje stvari: pametni hladilnik, pametno shrambo, pametne posode, pametno tehtnico, pametno ponev, pametni štedilnik. Poleg stvari pa še kuhinjski prehod in nabor spletnih storitev za pridobivanje recepta, pridobivanje podatkov iz stvari in za nadzor stvari (npr. nastavljanje temperature na

štedilniku, nastavljanje časovnikov). V tem primeru bo aplikacija IoT najbrž vsebovala tudi neke vrste zaslon za prikaz podatkov (npr. pametni telefon, tablica).

V nadaljevanju tega dela se bomo osredotočili predvsem na del aplikacije IoT, ki se nanaša na stvari. Za implementacijo storitev in podatkovne shrambe se lahko v veliki meri opremo na storitve v računalniškem oblaku, primerjavo le-teh pa lahko najdemo v drugih delih. Podobno velja tudi za razvoj uporabniških vmesnikov na polno povezanih napravah, ki prikazujejo podatke pridobljene s spletnih storitev.

## 2.5. Tehnologije IoT

V nadaljevanju bomo najprej predstavili komunikacijske protokole, ki jih lahko uporabimo v aplikacijah IoT. Potem bomo predstavili preglednico nekaterih strojnih platform za prototipiranje in implementacijo stvari.

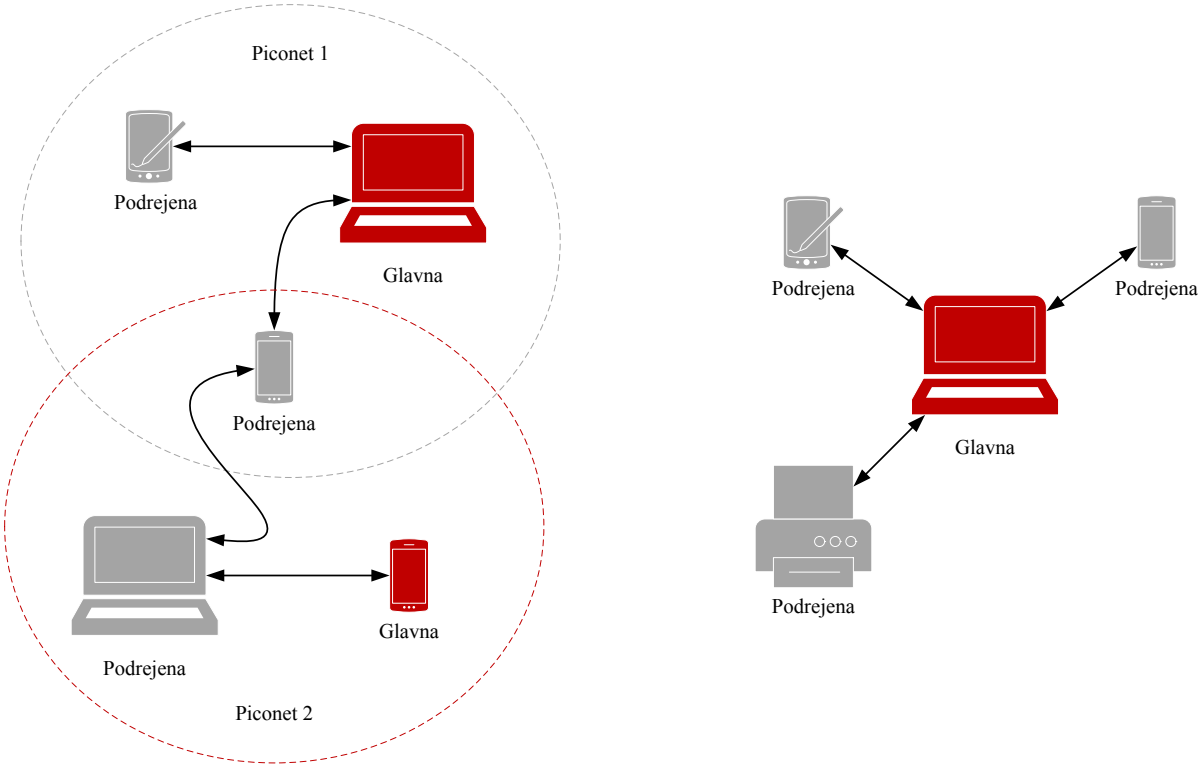
### 2.5.1. Komunikacijski protokoli

#### 2.5.1.1. *Bluetooth* čez IEEE 802.15.1

*Bluetooth* je bil zasnovan za brezžične povezave s kratkim dosegom, za omrežja WPAN (angl. *Wireless Personal Area Network*). V omrežjih Bluetooth se naprave povezujejo v topologijo imenovano *piconet* (Slika 10, levo). V topologiji *piconet* je ena od naprav *Bluetooth* glavna, druge pa se povezujejo nanjo in so ji podrejene. Podrejene naprave lahko komunicirajo le z eno glavno napravo, glavna naprava lahko komunicira tako s posamezno podrejeno napravo ali pa z več podrejenimi napravami hkrati. Več prekrivajočih se omrežij *piconet* imenujemo omrežje *scatternet* (Slika 10, levo). Posamezna naprava *Bluetooth* je lahko v več omrežjih *piconet* hkrati, vendar pa je lahko glavna v največ enem od njih. Podrejene naprave lahko preidejo v način varčevanja z energijo [30].

#### 2.5.1.2. BLE

Nizkoenergijski *Bluetooth* (BLE) je določen v specifikaciji *Bluetooth* 4.0. Podobno kot v omrežjih *Bluetooth* je lahko naprava BLE glavna ali podrejena. Podrejena naprava je lahko povezana na natanko eno glavno napravo, posamezna glavna naprava pa je lahko povezana z več podrejenimi napravami. Naprave v omrežjih BLE so torej povezane v zvezdasto topologijo (Slika 10, desno). Podrejene naprave oglašujejo prisotnost na treh vnaprej določenih kanalih. Glavne naprave preverjajo te kanale in tako odkrivajo nove podrejene naprave. Komunikacija poteka v povezavnih dogodkih (angl. *connection events*), kjer se glavna in podrejena naprava sinhronizirano zbudita in izmenjata podatke [31, 32].




*Slika 10. Scatternet, levo in zvezdasta topologija, desno.*

#### 2.5.1.3. UWB čez IEEE 802.15.3

UWB (angl. *ultra-wideband*) omogoča hitrosti prenosa do 480 Mbit/s in je primeren za multimedijske aplikacije v domačih omrežjih. Zaradi težav z regulacijo radijskega spektra tako v Evropi kot v ZDA protokol trenutno ni široko razširjen [30].

#### 2.5.1.4. ZigBee čez IEEE 802.15.4

ZigBee je standard za počasna omrežja WPAN (angl. *low rate* WPAN) v katera se povezujejo naprave z minimalno porabo energije. Naprave ZigBee se lahko organizirajo v zankasta omrežja, ki se sama organizirajo tako, da je vsaka naprava dostopna v nekaj skokih (angl. *multi hop*). Naprava ZigBee se lahko v omrežje poveže kot:

- polnovredna naprava (angl. *full function device*, FFD). Naprave FFD so lahko koordinatorji omrežij ali pa usmerjevalniki. Naprave FFD lahko komunicirajo z drugimi FFD napravami ali pa z RFD napravami.
- naprava z omejeno zmogljivostjo (angl. *reduced function device*, RFD). Naprave RFD so lahko povezane samo na eno napravo FFD.

Omrežja *ZigBee* imajo zvezdasto topologijo (Slika 10, dresno) [30].

#### 2.5.1.5. Wi-Fi čez IEEE 802.11 a/b/g

Družina standardov Wi-Fi je namenjena brezžičnim lokalnim omrežjem (angl. *wireless local area network*, WLAN). V omrežjih Wi-Fi so lahko naprave povezani infrastrukturno topologijo z odjemalci in eno ali več dostopnimi točkami, ali pa v topologijo ad-hoc, kjer vsaka naprava sodeluje pri usmerjanju in posredovanju podatkov [30].

#### 2.5.1.6. Primerjava

V Tabela 1 smo zbrali lastnostni predstavljenih brezžičnih tehnologij, ki nas zanimajo pri načrtovanju modela aplikacij IoT.

*Tabela 1. Primerjava brezžičnih tehnologij.*

| | Bluetooth | BLE | UWB | ZigBee | Wi-Fi |
|---|---|---|---|---|---|
| Frekvenca | 2,4 GHz | 2,4 GHz | 3,1–10,6 GHz | 868/915 MHz | 2,4 GHz<br>5 GHz |
| Teoretični največji prenos | 1 Mbit/s | 1 Mbit/s | 110 Mbit/s | 250 kbit/s | 54 Mbit/s |
| Omrežje enak z enakim | ne | ne | ne | da | da |
| Šifriranje | E0-stream | AES | AES | AES | AES |
| Avtentikacija | Skupen ključ | CBC-MAC | CBC-MAC | CBC-MAC | CCMP |
| Celovitost | CRC-16 | CRC-24 | CRC-32 | CRC-16 | CRC-32 |
| Poraba Tx [mA] | 57 | 15 | 227 | 25 | 219 |
| Poraba Rx [mA] | 47 | 15 | 227 | 27 | 215 |
| Domet [m] | <100 | <100 | <10 | <10 | <35 |

### 2.5.2. Strojne platforme

V Tabela 2 in Tabela 3 so zbrani podatki za nekaj strojnih platform IoT. Med izbranimi platformami so vsi procesorji ARM in segajo od 4 megaherčnih eno jedrnih procesorjev do gigaherčnih več jedrnih. Proizvajalci platform, ki jih lahko uporabimo za IoT, tekmujejo, kdo bo ponudil več procesorske zmogljivosti, več digitalnih in več analognih vmesnikov za čim manjšo ceno. Kot zanimivost izpostavimo še skok zmogljivosti – za 2x višjo ceno, dobimo več kot 2x bolj zmogljivo platformo. Seveda pa to pomeni samo, da ni vse v zmogljivostih. V aplikacijah IoT je večkrat pomembnejša poraba energije [33].

*Tabela 2. Strojne platforme.*

| | PanStamps | TinyDuino | Arduino Uno | RFduino | XinoRF |
|---|---|---|---|---|---|
| Procesor | atmega328p | atmega328p | atmega328 | cortex-m0 | atmega328 |
| RAM [kB] | 4 | 2 | 2 | 8 | 2 |
| Flash [kB] | 32 | 32 | 32 | 128 | 32 |
| Ura | 8 MHz | 4 MHz | 16 MHz | 16 MHz | |
| Brezžični vmesnik | TI CC1101 | | | BLE | SRF-U |
| Cena | $18,95 | $29,95 | $21 | $35 | $36 |
| Odprtost | GNU GPLv2 | da | da | da | da |
| Digitalni vhodi | 6 | 14 | 14 | 7 | 6 |
| Analogni vhodi | 6 | 6 | 6 | 4 | 6 |

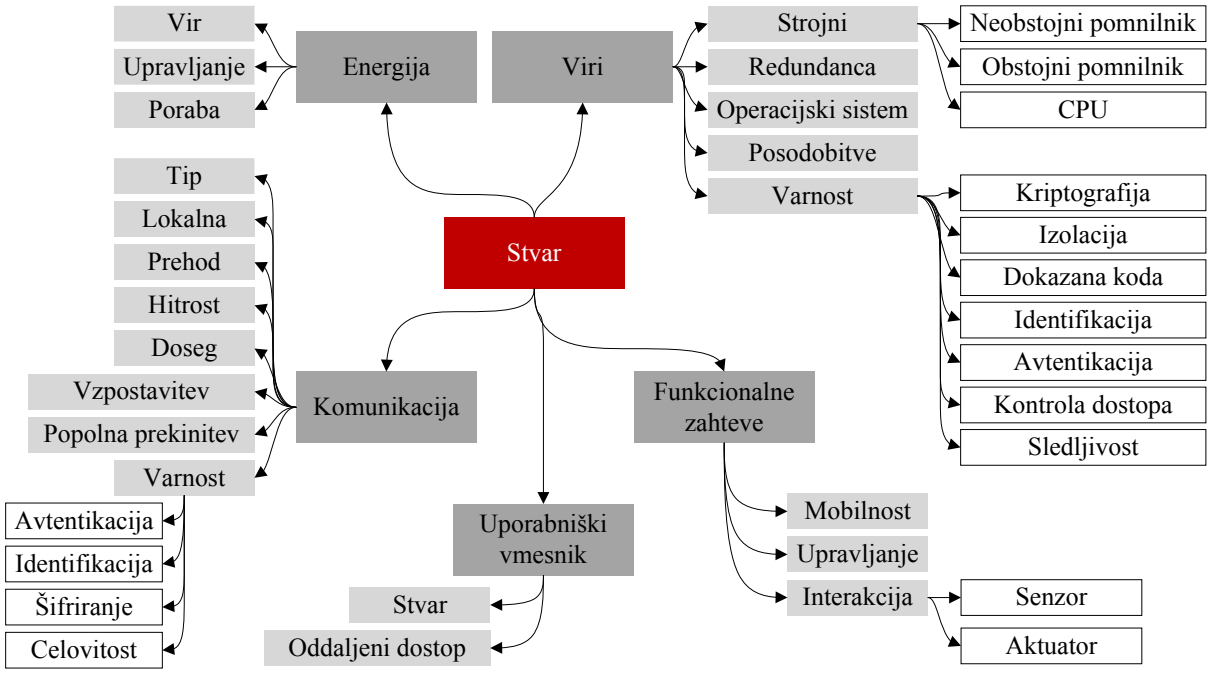
*Tabela 3. Strojne platforme.*

| | Ciseco | Pinoccio | Raspberry Pi | BeagleBone Black | CubieBoard |
|---|---|---|---|---|---|
| Procesor | atmel328 | atmega256rfr2 | bcm2835 | cortex-a8 | cortex-a8 |
| RAM [kB] | 32 | 32 | 512 x 2E10 | 512 x 2E10 | 512 x 2E10 |
| Flash [kB] | | | 32 | 2048 x 2E10 | 32 |
| Ura | 8 MHz | 16 MHz | 700 MHz | 1000 MHz | 1000 MHz |
| Brezžični vmesnik | GSM, XRF, RN-XV | ZigBee, WiFi | Wi-Fi | Wi-Fi | Wi-Fi |
| Cena | >$50 | $35 | $45 | $49 | $56 |
| Odprtost | da | da | delno | delno | da |
| Digitalni vhodi | | 17 | 26 | 2x46 | 96 |
| Analogni vhodi | | 8 | | | 6 |

# 3. Modeliranje aplikacij IoT

## 3.1. Lastnosti stvari

Lastnosti stvari povzemamo po taksonomiji interneta stvari v [6]. Taksonomija je v osnovi zasnovana razširljivo, kar pomeni, da bomo lahko dodali lastnosti, ki jih potrebujemo za modeliranje aplikacij IoT. Nekatere lastnosti iz [6] bomo zaradi redundance izpustili, nekatere pa preimenovali, da bodo skladnejše s terminologijo uporabljeno v našem delu.




*Slika 11. Hierarhija lastnosti stvari.*

Lastnosti bomo razdelili na pet glavnih kategorij: energija, komunikacije, funkcionalne zahteve, uporabniški vmesnik in viri. Slika 11 prikazuje hierarhijo lastnosti razdeljenih na naštete kategorije. Posamezne lastnosti bomo podrobneje opisali v nadaljevanju.

### 3.1.1. Energija

| energija.vir | |
|---|---|
| **Lastnost** | Vir energije |
| **Opis** | Določa na kakšen način dobi stvar električno energijo potrebno za delovanje. Stvari lahko energijo dobijo iz *okolja*, npr. oznaka RFID dobi energijo iz elektromagnetnega valovanja, ki ga oddaja antena RFID, pametni zabojnik za odpadke pa iz sončnih celic nameščenih na njemu. Stvari lahko energijo dobijo iz lokalnega vira, ki je lahko *zamenljiv* ali pa *nezamenljiv*, npr. pametna tehtnica vsebuje baterijo, ki jo lahko napolnimo ali zamenjamo. Stvari so lahko priključene na *stalen* vir električne energije, npr. pametni hladilnik je ves čas priključen na električno omrežje. |
| **Zaloga vrednosti** | okolje, zamenljiv, nezamenljiv, stalen |

| energija.upravljanje | |
|---|---|
| **Lastnost** | Upravljanje energije |
| **Opis** | Določa kako bo stvar porabljala energijo in s tem čas delovanja stvari. Stvar je *navadno izklopljena*, če je privzeto v stanju v katerem ne porablja energije, vključi se občasno ali s signalom iz okolja, npr. oznaka RFID se vklopi, ko ulovi signal antene RFID. Če mora stvar v vsakem trenutku varčevati in porabiti čim manj energije je njen način upravljanja energije *nizka poraba*, npr. pametna posoda ves čas varčuje z energijo, odzove pa se na premike, spremembe teže in posluša ukaze iz kuhinjskega stičišča. Stvar je *vedno vklopljena*, če ji ni potrebno varčevati z energijo in lahko ves čas prosto uporablja vse svoje senzorje, aktuatorje in komunikacijske vmesnike, npr. pametni hladilnik ves čas nadzira pametne posode, posodablja podatke in sprejema nadzorne ukaze. |
| **Zaloga vrednosti** | navadno-izklopljen, nizka-poraba, vedno-vklopljen |

| energija.poraba | |
|---|---|
| **Lastnost** | Poraba energije |
| **Opis** | Določa povprečno porabo električne energije stvari pri normalni uporabi na časovno enoto. Privzeta enota je mAh, npr. 1080 mAh. |
| **Zaloga vrednosti** | >= 0 |

### 3.1.2. Komunikacije

| komunikacija.tip | |
|---|---|
| **Lastnost** | Tip komunikacije |
| **Opis** | Določa tip vmesnika po katerem stvar komunicira z omrežjem stvari ali a prehodom oz. stičiščem. Tip je *ožičen*, če je stvar fizično priključena v omrežje stvari, npr. Ethernet. Tip je *brezžičen*, če je stvar v omrežje stvari priključena brezžično, npr. 802.11a/b/g, BLE. |
| **Zaloga vrednosti** | brezžičen, ožičen |

| komunikacija.lokalna | |
|---|---|
| **Lastnost** | Lokalna komunikacija |
| **Opis** | Opredeljuje ali ima stvar vmesnik in sposobnost komuniciranja z omrežjem stvari. Lokalna komunikacija z omrežjem stvari tipično porabi manj energije, kot komunikacija z omrežjem IP. Omrežje stvari je lahko implementirano s tehnologijo BLE, RFID, ZigBee, ipd. |
| **Zaloga vrednosti** | da, ne |

| komunikacija.prehod | |
|---|---|
| **Lastnost** | Komunikacija čez prehod ali stičišče |
| **Opis** | Opredeljuje ali potrebuje stvar komunikacijski prehod ali stičišče preko katerega lahko komunicira z omrežjem IP. Zaradi varčevanja z energijo ima lahko stvar samo vmesnik za komunikacijo z omrežjem stvari, vendar pa za delovanje potrebuje tudi podatke s storitev v omrežju IP. |

| | |
|---|---|
| | V tem primeru mora biti v omrežju stvari prisoten prehod ali stičišče. Npr. pametna posoda ima le vmesnik BLE, preko tega komunicira s kuhinjskim prehodom, ta pa ji omogoča sprejemanje ukazov iz nadzorne spletne storitve in pošiljanje podatkov na storitev za hrambo podatkov. |
| **Zaloga vrednosti** | da, ne |

| **komunikacija.popolna_prekinitev** | |
|---|---|
| **Lastnost** | Popolna prekinitev komunikacij |
| **Opis** | Lastnost predstavlja zmožnost stvari, da popolnoma izključi komunikacijske vmesnike, med tem pa še naprej opravlja svoje osnovne funkcije. Gre za način varčevanja z energijo, npr. števec korakov lahko šteje korake, povezavo BLE pa vključi le občasno za posredovanje podatkov posredniku. |
| **Zaloga vrednosti** | da, ne |

| **komunikacija.vzpostavitev** | |
|---|---|
| **Lastnost** | Vzpostavitev komunikacije |
| **Opis** | Smer vzpostavitve komunikacijskega kanala. Komunikacijski kanal lahko vzpostavi *stvar*, kot posledico spremembe notranjega stanja, npr. števec korakov, ki periodično posreduje podatke posredniku. Če komunikacijski kanal vzpostavlja *posrednik*, mora stvar ves čas spremljati svoj komunikacijski vmesnik za morebitno povezavo, npr. oznaka RFID čaka na signal antene RFID. Stvar in posrednik sta lahko tudi enakovredna in lahko *oba* vzpostavita, npr. pametna posoda periodično pošilja podatke o svojem stanju kuhinjskem prehodu, hkrati pa lahko kuhinjski prehod kadarkoli pošlje ukaz pametni posodi. |
| **Zaloga vrednosti** | stvar, posrednik, oba |

| **komunikacija.varnost.avtentikacija** | |
|---|---|
| **Lastnost** | Komunikacijska varnost – avtentikacija |
| **Opis** | Proces avtentikacije omogoča, da se stvar in posrednik prepričata s kom komunicirata. Avtentikacija je lahko *obojestranska*. V tem primeru morata tako stvar kot posrednik dokazati svojo identiteto, npr. števec korakov uparimo s pametnim telefonom. Če je avtentikacija *enosmerna*, potem svojo identiteto dokazuje bodisi stvar, bodisi posrednik, npr. antena RFID dokaže svojo identiteto, posamezne oznake RFID na izdelkih pa ne. Varnost je *brez* avtentikacije, če niti stvari niti posredniku ni treba dokazati svoje identitete, npr. *iBeacon* oddajniki za navigacijo v notranjih prostorih. |
| **Zaloga vrednosti** | obojestranska, enosmerna, brez |

| **komunikacija.varnost.identifikacija** | |
|---|---|
| **Lastnost** | Komunikacijska varnost – identifikacija |
| **Opis** | Proces avtentikacije omogoča preverjanje pripadnosti skupini stvari ali posrednikov. Proces identifikacije pa omogoča razlikovanje posameznih stvari ali posrednikov med seboj. Podobno kot avtenkacija je lahko identifikacija *obojestranska*, *enosmerna* ali pa odsotna (*brez*). |
| **Zaloga vrednosti** | obojestranska, enosmerna, brez |

| **komunikacija.varnost.šifriranje** | |
|---|---|
| **Lastnost** | Komunikacijska varnost – šifriranje |
| **Opis** | Šifriranje v tem kontekstu omogoča ohranjanje zaupnosti podatkov med transportom oz. med komunikacijo. Velja omeniti, da odsotnost šifriranja ne pomeni nujno ranljivosti, saj so lahko podatki sami javni, npr. trenutna temperatura zraka na vremenski postaji. Vrednost lastnosti je bodisi *brez*, če šifriranja ne uporabljamo, bodisi ime algoritma, ki se uporablja za šifriranje. |
| **Zaloga vrednosti** | brez, des, 3des, aes, … |

| komunikacija.varnost.celovitost | |
|---|---|
| **Lastnost** | Komunikacijska varnost – celovitost |
| **Opis** | Šifriranje nam omogoča ohranjanje zaupnosti podatkov, integriteta pa pomeni ohranjanje celovitosti podatkov. Podatek ostane celovit, če se med komunikacijo ni spremenil. Lastnost nam pove, če stvar ali posrednik pri komuniciranju preverjata celovitost podatkov, npr. števec korakov uporablja za komunikacijo BLE, ki ima vgrajeno preverjanje celovitosti. |
| **Zaloga vrednosti** | da, ne |

| komunikacija.hitrost | |
|---|---|
| **Lastnost** | Največja hitrost komunikacije |
| **Opis** | Teoretična največja hitrost prenosa podatkov s katero lahko komunikacijski vmesnik pošilja podatke, npr. naprave BLE imajo teoretično pasovno širino 1 Mbit/s. Če ne bomo zapisali drugače, bomo za to lastnost privzeli enoto kbit/s. |
| **Zaloga vrednosti** | >= 0 |

| komunikacija.doseg | |
|---|---|
| **Lastnost** | Največji doseg komunikacije |
| **Opis** | Teoretično največja razdalja, ki jo lahko podatki dosežejo brez popačenja pri uporabi dane tehnologije, npr. BLE ima doseg do 100 metrov. Če ne bomo zapisali drugače, bomo za to lastnost privzeli enoto m (meter). |
| **Zaloga vrednosti** | >= 0 |

### 3.1.3. Funkcionalne zahteve

| **funkcija.interakcija.senzor** | |
|---|---|
| **Lastnost** | Senzor za zajem podatkov iz okolja |
| **Opis** | Stvar ima lahko senzor s katerim zazna in zajame podatke it svojega okolja, npr. trenutno temperaturo. Senzor ima lahko lokalni *pomnilnik* v katerega lahko shrani nekaj meritev dokler ne bo stvar povezana s posrednikom, npr. števec korakov ves čas šteje korake, podatke pa periodično prenese na pametni telefon. Stvar ima lahko tudi senzor brez pomnilnika (*da*), kar pomeni, da mora konstantno pošiljati podatke posredniku, vrednost senzorja pa se sproti posodablja, npr. senzor za spremljanje temperature ozračja. Stvar je seveda lahko tudi brez senzorja (*ne*). |
| **Zaloga vrednosti** | pomnilnik, da, ne |

| **funkcija.interakcija.aktuator** | |
|---|---|
| **Lastnost** | Aktuator za manipulacijo z okoljem |
| **Opis** | Stvar ima lahko aktuator s katerim lahko manipulira v svojem okolju, npr. premakne predmet, odda svetlobo, vklopi stikalo. Za primer lahko vzamemo pametno posodo, ki obarva diodo LED rdeče, če se je hrana v posodi pokvarila. |
| **Zaloga vrednosti** | da, ne |

| **funkcija.mobilnost** | |
|---|---|
| **Lastnost** | Mobilnost stvari |
| **Opis** | Stvari so lahko *fiksne*, kar pomeni, da se večinoma nahajajo na isti lokaciji ali pa se minimalno premikajo v omejenem prostoru, npr. vremenska postaja. Druge stvari so *mobilne* in načrtovane za konstantno gibanje, npr. števec korakov. |
| **Zaloga vrednosti** | fiksna, mobilna |

| funkcija.upravljanje | |
|---|---|
| **Lastnost** | Upravljanje stvari |
| **Opis** | Nekaterih stvari ni potrebno upravljati (*brez*), npr. oznake RFID. Določene stvari pa zahtevajo konfiguracijo, kar pomeni, da jih moramo upravljati. Glede na to kako lahko stvari upravljamo ločimo *lokalno* upravljanje, npr. uparjanje števca korakov s telefonom; in *oddaljeno* upravljanje, ker lahko stvar nastavimo skozi komunikacijski vmesnik, npr. nastavimo katera hrana se nahaja v naši pametni posodi. |
| **Zaloga vrednosti** | brez, lokalna, oddaljena |

### 3.1.4. Uporabniški vmesnik

| ui.stvar | |
|---|---|
| **Lastnost** | Uporabniški vmesnik stvari |
| **Opis** | Stvari imajo lahko komponente, ki omogočajo interakcijo z uporabniki. Uporabniški vmesnik je lahko *aktiven*, kar pomeni, da so deli svari namenjeni interakciji z uporabnikom, npr. stvar za spremljanje teka ima gumb s katerim začnemo tek. Uporabniški vmesnik je lahko *pasiven*, kar pomeni, da uporabnik ne more neposredno uporabljati stvari, stvar pa lahko posredno komunicira z uporabnikom, npr. preko ekrana, z zvokom, svetlobo, vibracijami. Primer pasivnega uporabniškega vmesnika je pripomoček za navigacijo na kolesu, ki prikazuje puščico v smer v katero moramo zaviti. Stvar ima lahko tudi kombinacijo aktivnega in pasivnega uporabniškega vmesnika (*oboje*), npr. aplikacija na pametnem telefonu. |
| **Zaloga vrednosti** | aktiven, pasiven, brez |

| ui.oddaljen_dostop | |
|---|---|
| **Lastnost** | Oddaljeni dostop do uporabniškega vmesnika |
| **Opis** | Včasih lahko do uporabniškega vmesnika stvari pridemo preko posrednika v omrežju stvari, npr. določimo vsebino pametne posode preko spletnega vmesnika. |
| **Zaloga vrednosti** | da, ne |

### 3.1.5. Viri

| viri.strojni.neobstojni_pomnilnik | |
|---|---|
| **Lastnost** | Neobstojni pomnilnik RAM |
| **Opis** | Količina neobstojnega (angl. *volatile*) pomnilnika, tudi delovnega pomnilnika, ki ga imamo na voljo na stvari. Če ne bomo določili drugače, privzemimo enoto kB. |
| **Zaloga vrednosti** | >= 0 |

| viri.strojni.obstojni_pomnilnik | |
|---|---|
| **Lastnost** | Obstojni pomnilnik |
| **Opis** | Količina obstojnega (angl. non-volatile), tudi trajnega, pomnilnika, ki nam je na voljo na stvari. Če ne bomo določili drugače, privzemimo enoto kB. |
| **Zaloga vrednosti** | >= 0 |

| viri.strojni.cpu | |
|---|---|
| **Lastnost** | Tip procesorja |
| **Opis** | Tip glavnega procesorja, ki ga uporablja stvar. Vrednosti predstavljajo različne tipe procesorjev, npr. platforma BeagleBone uporablja procesorje Cortex-A8 |
| **Zaloga vrednosti** | atom_n435, cortex-m, avr, cortex-a8, i7-3820, … |

| viri.varnost.kriptografija | |
|---|---|
| **Lastnost** | Strojno podprta kriptografija |
| **Opis** | Strojno podprta kriptografija pospeši kriptografske operacije in lahko prihrani pri energiji, če jo stvar omogoča. Sistemi, ki uporabljajo BLE, imajo strojno podprte kriptografske operacije potrebne za komunikacijo BLE. |
| **Zaloga vrednosti** | da, ne |

| viri.varnost.izolacija | |
|---|---|
| **Lastnost** | Izolacija procesov |
| **Opis** | Stvar omogoča zagon procesov v izolaciji, npr. privzeto razvojno okolje za Arduino ne omogoča izolacije procesov. |
| **Zaloga vrednosti** | da, ne |

| viri.varnost.dokazana_koda | |
|---|---|
| **Lastnost** | Dokazana koda |
| **Opis** | Programska koda, ki teče na stvari je dokazana in stvar se bo obnašala po pričakovanjih in na deterministični način. |
| **Zaloga vrednosti** | da, ne |

| viri.varnost.identifikacija | |
|---|---|
| **Lastnost** | Identifikacija uporabnika |
| **Opis** | Če stvar omogoča interakcijo z uporabnikom, bo pred interakcijo identificirala uporabnika in preverila uporabnikovo identiteto. Pravzaprav imamo tudi tukaj enake možnosti za identifikacijo, kot pri komunikacija.varnost.identifikacija: *obojestranska*, *enosmerna, brez*. Npr. pametni telefon identificira uporabnika, ko ta vnese številko PIN. |
| **Zaloga vrednosti** | obojestranska, enosmerna, brez |

| viri.varnost.avtentikacija | |
|---|---|
| **Lastnost** | Avtorizacija uporabnika |
| **Opis** | Če stvar omogoča interakcijo z uporabnikom, potem bo pravilno implementirana avtorizacija poskrbela, da ima uporabnik dostop samo do stvari in funkcionalnosti za katere ima dodeljene pravice. Avtorizacija je podobno kot komunikacija.varnost.identifikacija lahko: *obojestranska*, *enosmerna, brez*. Sistem omaric za shranjevanje oblačil na kopališču, bo z enosmerno avtorizacijo uporabniku odprl pravo (uporabnikovo) omarico. |
| **Zaloga vrednosti** | obojestranska, enosmerna, brez |

| viri.varnost.kontrola_dostopa | |
|---|---|
| **Lastnost** | Kontrola in upravljanje oddaljenega dostopa do virov |
| **Opis** | Če stvar omogoča oddaljen dostop do virov (senzorjev, aktuatorjev, podatkov) iz omrežja stvari, potem lastnost določa ali je možno omejiti dostop do posameznega vira. Npr. napredni sistem za lociranje predmetov omogoča, da podatkovne mule zaznajo prisotnost predmeta in to sporočijo spletni storitve, ne morejo pa dostopati do drugih podatkov na predmetu. |
| **Zaloga vrednosti** | da, ne |

| viri.varnost.sledljivost | |
|---|---|
| **Lastnost** | Sledljivost aktivnosti |
| **Opis** | Stvar beleži svoje aktivnosti in tako omogoča sledljivost. Npr. pametni hladilnik beleži kdo ga je odpiral in končno omogoči, da najdemo zlonamerneža, ki nam je pojedel sendvič. |
| **Zaloga vrednosti** | da, ne |

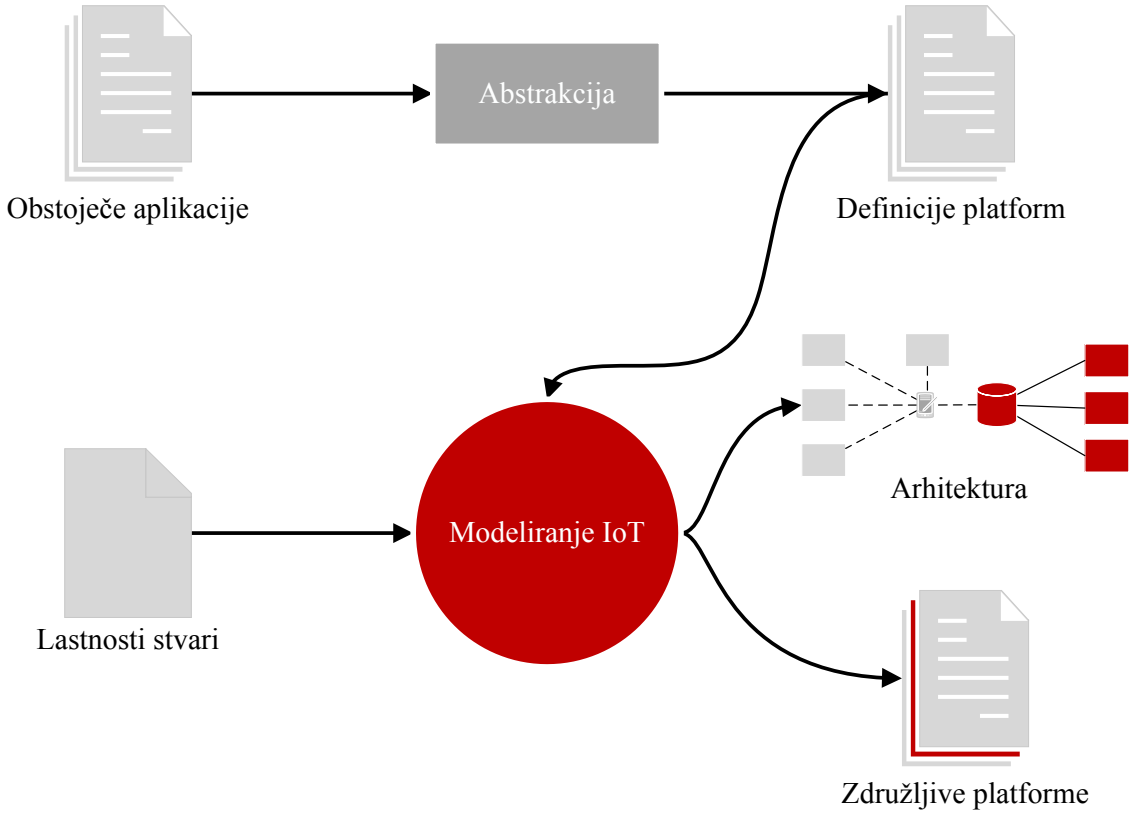
| viri.redundanca | |
|---|---|
| **Lastnost** | Zagotavljanje pravilnosti podatkov |
| **Opis** | Stvar ima lahko več redundantnih načinov za preverjanje istega stanja in s tem zagotavlja večjo pravilnost podatkov. Npr. pametni hladilnik lahko zazna to, da je odprt s svetlobnim senzorjem in z gumbom, ki se pritisne, ko zapremo vrata. |
| **Zaloga vrednosti** | da, ne |

| viri.os | |
|---|---|
| **Lastnost** | Operacijski sistem |
| **Opis** | Operacijski sistem, ki teče na stvari. Operacijski sistem je lahko implementiran s programsko in strojno opremo (*hwsw*), npr. na platformi Arduino. Operacijski sistem je implementiran kot jedro in strojna oprema (*jedro*). Operacijski sistem je v celoti implementiran v strojni opremi (*hw*), npr. v oznakah RFID. |
| **Zaloga vrednosti** | hwsw, jedro, hw |

| viri.posodobitve | |
|---|---|
| **Lastnost** | Posodobitve programske opreme |
| **Opis** | Posodobimo lahko programsko opremo, ki teče na svari, npr. platforma RFduino omogoča posodobitve programske opreme prek BLE. |
| **Zaloga vrednosti** | da, ne |

## 3.2. Postopek modeliranja

Slika 12 prikazuje postopek modeliranja aplikacije IoT. Postopek modeliranja začnemo s pripravo modela, nadaljujemo s pripravo zahtev, na koncu pa poženemo program za modeliranje, ki identificira možne arhitekture in platforme za izvedbo naše aplikacije IoT.




*Slika 12. Postopek modeliranja aplikacije IoT.*

### 3.2.1. Priprava modela

Pripravo modela začnemo s pripravo podatkov o obstoječih aplikacijah IoT. Za vsako aplikacijo pripravimo seznam lastnosti, kot smo ga določili v poglavju 3.1. Primer seznama lastnosti za stvar *Fitbit One* [19] prikazuje Izpis 1. Seznam je zapisan v obliki JSON, ki ga lahko uporabimo v našem programu za modeliranje aplikacij IoT.

```
{
    "energija.vir": "zamenljiv",
    "energija.upravljanje": "nizka-poraba",
    "energija.poraba": 10,
    "komunikacija.tip": "brezžičen",
    "komunikacija.lokalna": "da",
    "komunikacija.prehod": "da",
    "komunikacija.popolna_prekinitev": "da",
    "komunikacija.vzpostavitev": "stvar",
    "komunikacija.varnost.avtentikacija": "obojestranska",
    "komunikacija.varnost.identifikacija": "obojestranska",
    "komunikacija.varnost.šifriranje": "aes-ccm",
    "komunikacija.varnost.celovitost": "ne",
```

```
    "komunikacija.hitrost": 1000,
    "komunikacija.doseg": 100,
    "funkcija.interakcija.senzor": "pomnilnik",
    "funkcija.interakcija.aktuator": "da",
    "funkcija.mobilnost": "mobilna",
    "funkcija.upravljanje": "brez",
    "ui.stvar": [ "aktiven", "pasiven" ],
    "ui.oddaljen_dostop": "da",
    "viri.strojni.neobstojni_pomnilnik": 50,
    "viri.strojni.obstojni_pomnilnik": 250,
    "viri.strojni.cpu": "stm32l",
    "viri.varnost.kriptografija": "da",
    "viri.varnost.izolacija": "ne",
    "viri.varnost.dokazana_koda": "ne",
    "viri.varnost.identifikacija": "ne",
    "viri.varnost.avtentikacija": "ne",
    "viri.varnost.kontrola_dostopa": "ne",
    "viri.varnost.sledljivost": "ne",
    "viri.redundanca": "ne",
    "viri.os": "swhw",
    "viri.posodobitve": "da"
}
```

*Izpis 1. Seznam lastnosti v obliki JSON.*

V procesu abstrakcije združimo zbrane podatke o obstoječih aplikacijah IoT v definicije platform. Gradimo seznam vseh videnih vrednosti za vsako lastnost na vsaki platformi. Rezultat je seznam lastnosti, ki so podprte na posameznih platformah.

### 3.2.2. Zajem zahtev

Za vsako od stvari, ki jih bomo uporabili v naši aplikaciji IoT pripravimo seznam lastnosti, ki jih stvar potrebuje za delovanje. Uporabimo seznam lastnosti stvari, ki smo ga določili v 3.1. Pazimo le, da pri številčnih podatkih vnesemo minimalne zahteve (namesto teoretičnih zmogljivosti). Za lastnost komunikacija.hitrost bomo na primer navedli oceno hitrosti (10 kbit/s), ki jo potrebujemo za normalno delovanje, in ne teoretične zmogljivosti tehnologije BLE (1000 kbit/s).

Izpis 2 prikazuje primer zahtev za pametni hladilnik. Za lastnost komunikacija.tip smo vnesli znak *. S tem povemo programu za modeliranje, da nam v končni rešitvi ustreza katerakoli vrednost te lastnosti.

```
{
    "energija.vir": "stalen",
    "energija.upravljanje": "vedno-vklopljen",
    "energija.poraba": 1363,
    "komunikacija.tip": "*",
    "komunikacija.lokalna": "ne",
    "komunikacija.prehod": "ne",
    "komunikacija.popolna_prekinitev": "ne",
    "komunikacija.vzpostavitev": "oba",
    "komunikacija.varnost.avtentikacija": "obojestranska",
    "komunikacija.varnost.identifikacija": "obojestranska",
    "komunikacija.varnost.šifriranje": "da",
    "komunikacija.varnost.celovitost": "da",
    "komunikacija.hitrost": 1000,
    "komunikacija.doseg": 20,
    "funkcija.interakcija.senzor": "pomnilnik",
    "funkcija.interakcija.aktuator": "da",
    "funkcija.mobilnost": "fiksna",
    "funkcija.upravljanje": [ "lokalna", "oddaljena"],
    "ui.stvar": [ "aktiven", "pasiven" ],
    "ui.oddaljen_dostop": "da",
    "viri.strojni.neobstojni_pomnilnik": 8000000,
    "viri.strojni.obstojni_pomnilnik": 60000000,
    "viri.strojni.cpu": "*",
    "viri.varnost.kriptografija": "da",
    "viri.varnost.izolacija": "ne",
    "viri.varnost.dokazana_koda": "ne",
    "viri.varnost.identifikacija": "ne",
    "viri.varnost.avtentikacija": "ne",
    "viri.varnost.kontrola_dostopa": "da",
    "viri.varnost.sledljivost": "da",
    "viri.redundanca": "ne",
    "viri.os": "swhw",
    "viri.posodobitve": "da"
}
```

*Izpis 2. Seznam zahtevanih lastnosti za pametni hladilnik.*

### 3.2.3. Rezultat

Za vsako od stvari, za katero smo 3.2.2 identificirali zahteve, poiščemo platforme, ki omogočajo njihovo izvedbo. S statično določenim naborom pravil identificiramo mogoče arhitekture za izvedbo.

Na ta način dobimo za vsako od stvari arhitekturo ali nabor arhitektur in za vsako arhitekturo nabor platform, ki omogočajo izvedbo.

Na zadnjem koraku, če je potrebno, združimo arhitekture in poiščemo minimalen nabor platform, ki nam omogoča izvedbo naše aplikacije IoT. Primer rezultata modela za pametnih hladilnik je prikazan na Izpis 3.

```
Arhitektura: 2.2.5
Platforme: i5-6260u, cortex-a7, cortex-a8
```

*Izpis 3. Primer izpisa rezultata modela.*

# 4. Aplikacija

Postavimo se v vlogo mladega inženirja, ki ga pesti praktična težava. Vedno, ko kakršnokoli hrano spravi v plastično posodo, ki jo postavi v hladilnik, nanjo eventualno pozabi in nekaj tednov kasneje je soočen z neprijetnim čiščenjem plastične posode in spoznanjem, da na tem svetu zavržemo preveč hrane. V enem od tovrstnih pretresljivih trenutkov se mladi inženir odloči, da je rešitev v izdelavi pametne posode. Pametna posoda bo spreminjala barvo glede na to koliko časa bo v hladilniku. Prvi dan bo zelena, drugi in tretji dan rumena, četrti dan pa bo postala rdeča. Hkrati naj bo posoda opremljena še s tehtnico, čisto za vsak slučaj, da se ne bi mladi inženir nadejal večerje iz pametne posode, katere vsebino je njegov sostanovalec pojedel že za kosilo.

## 4.1. Model aplikacije IoT

Najprej podajmo zahteve za aplikacijo IoT. Posoda, ki smo si jo zamislili bo energijo črpala iz baterije. Baterije ne želimo prepogosto menjati ali polniti, zato postavimo zahtevo za nenehno upravljanje porabe energije, da bo le ta minimalna. To je naša prva aplikacija IoT, zato ne znamo oceniti kakšna bo povprečna poraba energije, lastnosti `energija.poraba` pustimo da zasede poljubno vrednost. Zaradi varčevanja z energijo se ne nadejamo povezljivosti v omrežje Internet. Lokalna povezljivost bo zadostovala, pa čeprav to pomeni, da bomo verjetno morali implementirati še posrednika.

Pametna posoda bo imela senzor – tehtnico in aktuator – LED diode z možnostjo prilagajanja barve. Pametna posoda lahko kadarkoli dobi zahtevo oz. povpraševanje po trenutni teži, ali pa ukaz, da naj prižge diodo LED. Posoda naj sicer tudi sama periodično sporoča svojo težo in stanje diode LED posredniku.

Med tem, ko želimo varno in avtenticirano komunikacijo med prehodom in pametno posodo (da nam ne bi sostanovalec nagajal z lažnim poročanjem, da je v hladilniku 5 pametnih posod, vsaka z več kot dovolj hrane), pa uporabnika ne bomo avtenticirali, saj pametna posoda ni opremljena s senzorji, ki bi omogočali prepoznavanje uporabnika (lastnosti `viri.varnost.avtentikacija`, `viri.varnost.avtentikacija`).

```
{
    "energija.vir": "zamenljiv",
    "energija.upravljanje": "nizka-poraba",
    "energija.poraba": "*",
    "komunikacija.tip": "brezžičen",
    "komunikacija.lokalna": "da",
    "komunikacija.prehod": "da",
    "komunikacija.popolna_prekinitev": "da",
    "komunikacija.vzpostavitev": "oba",
    "komunikacija.varnost.avtentikacija": "obojestranska",
    "komunikacija.varnost.identifikacija": "obojestranska",
    "komunikacija.varnost.šifriranje": "da",
    "komunikacija.varnost.celovitost": "da",
    "komunikacija.hitrost": 10,
    "komunikacija.doseg": 20,
    "funkcija.interakcija.senzor": "pomnilnik",
    "funkcija.interakcija.aktuator": "da",
    "funkcija.mobilnost": "mobilna",
    "funkcija.upravljanje": [ "lokalna", "oddaljena"],
    "ui.stvar": "pasiven",
    "ui.oddaljen_dostop": "da",
    "viri.strojni.neobstojni_pomnilnik": 128,
    "viri.strojni.obstojni_pomnilnik": "*",
    "viri.strojni.cpu": "*",
    "viri.varnost.kriptografija": "da",
    "viri.varnost.izolacija": "*",
    "viri.varnost.dokazana_koda": "*",
    "viri.varnost.identifikacija": "ne",
    "viri.varnost.avtentikacija": "ne",
    "viri.varnost.kontrola_dostopa": "da",
    "viri.varnost.sledljivost": "*",
    "viri.redundanca": "ne",
    "viri.os": "*",
    "viri.posodobitve": "da"
}
```

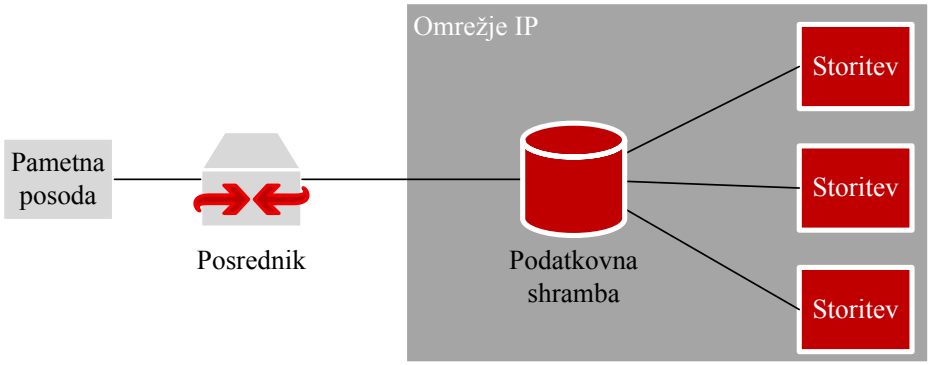
*Izpis 4. Seznam zahtevanih lastnosti za pametno posodo.*

Program za modeliranje IoT nam vrne rezultat, prikazan na Izpis 5, in predlaga arhitekturo prikazano na Slika 13. Na tem mestu priznamo, da nas je rezultat na Izpis 5 presenetil. Glede na podatke, uporabljene za pripravo modela, smo pričakovali več predlaganih platform – vsaj še cortex-a7, cortex-a8, atmega328, atmega2560. Med podrobnejšo analizo smo ugotovili, da

je rezultat kljub vsemu pravilen, saj je platforma rfduino res edina platforma, ki omogoča brezžično komunikacijo v omrežju IoT. Drugim Arduino platformam pa moramo za povezovanje v omrežje stvari dodati modul s komunikacijskih vmesnikom BLE.

```
Arhitektura: 2.2.3
Platforme: rfduino
```

*Izpis 5. Rezultat programa za modeliranje IoT.*




*Slika 13. Predlagana arhitektura aplikacije IoT – pametne posode.*

V nadaljevanju bomo predstavili nekatere implementacijske podrobnosti aplikacije IoT za pametno posodo. Kljub temu, da posrednik oz. prehod in storitve v omrežju IP niso del našega modela, smo jih vseeno implementirali za dokaz koncepta (angl. proof of concept).

## 4.2. Pametna posoda

Predlagana platforma rfduino je vgradna platforma, zasnovana na mikrokrmilniku Nordic Semiconductor nRF51822. Mikrokrmilnik združuje procesor ARM Cortex-M0, 256 kB obstojnega pomnilnika in 32 kB delovnega pomnilnika RAM. Posebnost arhitekture pa je vgrajen krnilnik Bluetooth Low Energy. Vse skupaj je zapakirano ohišje, ki je komajda večje od priključka USB-A.

Platforma rfduino je združljiva s plaftormami Arduino, zato lahko programe zanjo razvijamo v razvojnem okolju Arduino.

Programi za Arduino so v grobem razdeljeni na dva dela oz. dve funkciji: inicializacijo `setup()` in glavno zanko `loop()`. Med inicializacijo nastavimo delovanje naprave, v glavni zanki pa ustrezno spreminjamo notranje stanje stvari in poskrbimo za podatke s senzorjev. Zmožnosti komunikacije BLE dodajo še posebne funkcije, ki se pokličejo ob nastopu določenih dogodkov BLE: pri vzpostavitvi nove povezave `SimbleeBLE_onConnect`, pri prekinitivi povezave `SimbleeBLE_onDisconnect` in pri sprejemu novega sporočila

`SimbleeBLE_onReceive`. V nadaljevanju se bomo osredotočili predvsem na programsko kodo za komunikacijo s stvarjo.

### 4.2.1. Inicializacija sklada BlueTooth

Poglejmo si najprej inicializacijo sklada BlueTooth (Izpis 6). `SimbleeBLE.deviceName` je ime stvari na podlagi katerega bomo lahko stvar avtenticirali (pripadnost skupini). Stvar pa bomo lahko identificirali na podlagi njenega naslova MAC. S parametrom `SimbleeBLE.advertisementInterval` določimo kako pogosto bo stvar oznanila svojo prisotnost v omrežju stvari. Daljši interval (podan je v milisekundah) je ugodnejši s stališča porabe energije, vendar pa njegovo podaljšanje poslabša tudi latenco povezave.

```
    SimbleeBLE.deviceName = "PametnaPosoda";
    SimbleeBLE.advertisementInterval = 1000;
    SimbleeBLE.connectable = true;
    SimbleeBLE.begin();
```

*Izpis 6. Inicializacija sklada BlueTooth.*

### 4.2.2. Sprejem sporočila

Izpis 7 prikazuje kako se odzovemo na sprejem sporočila. Vidimo, da pametna posoda razume naslednje ukaze:

- png: ukaz ping, omogoča testiranje povezljivosti.
- ack: pametna posoda pošlje potrditev sekvenčne številke.
- on: prižge diodo LED v barvi, ki je podana kot parameter.
- off: ugasne diodo LED.
- wgh: vrne trenutno težo pametne posode.
- tar: ponastavi trenutno težo tehtnice na 0g (tara).

```
void SimbleeBLE_onReceive(char *buf, int len) {
    if (len >= 3 && Util::cmpBytes(buf, "png", 3)) {
        queue.PushPong();
    } else if (len == 9 && Util::cmpBytes(buf, "ack", 3)) {
        queue.Ack(buf);
    } else if (len == 13 && Util::cmpBytes(buf, "on ", 3)) {
        leds.On(&buf[3], Util::char2ulint(&buf[9]));
    } else if (len == 3 && Util::cmpBytes(buf, "off", 3)) {
        leds.Off();
    } else if (len == 3 && Util::cmpBytes(buf, "wgh", 3)) {
        scale.Read();
        queue.PushWeight(scale.GetWeight());
```

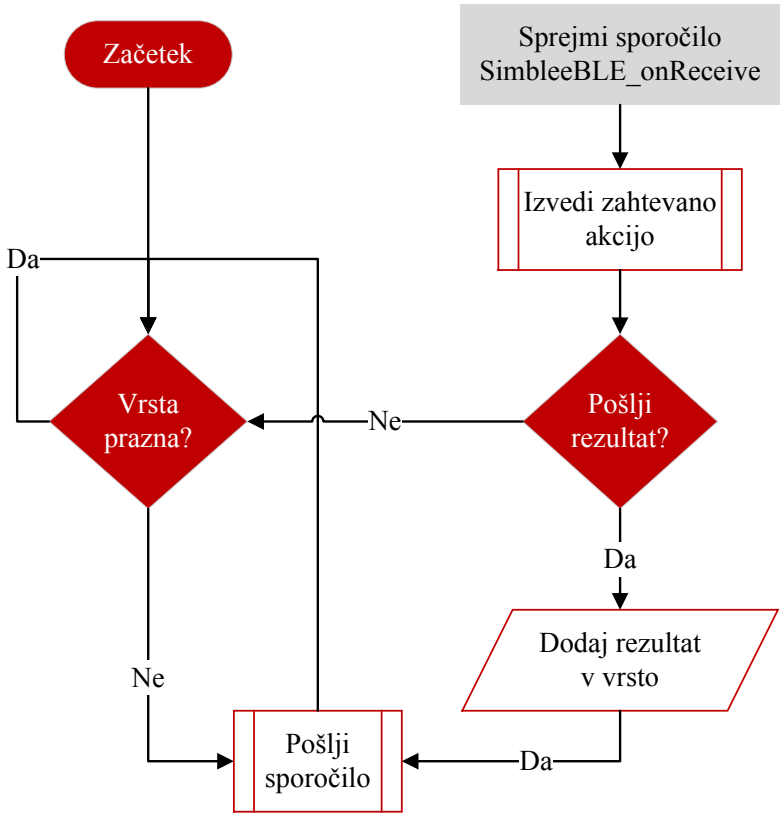
```
    } else if (len == 3 && Util::cmpBytes(buf, "tar", 3)) {
        scale.Tare();
    }
    queue.Send();
}
```

*Izpis 7. Sprejem prihajajočega podatka.*

Pozornem bralcu ne bo ušlo, da za pošiljanje uporabljamo vrsto in da uporabljamo sekvenčne številke. Razlog za to je v omejitvah protokola in implementacije BLE za izbran mikrokrmilnik. Funkcija `SimbleeBLE.send` deluje tako, da naše sporočilo nastavi kot eno od karakteristik, ki jo potem začne oglaševati. Zgodi se lahko, da želimo novo sporočilo poslati še preden je posrednik prejel trenutno sporočilo. Težavo smo rešili s potrjevanjem.

### 4.2.3. Protokol za izmenjavo sporočil

Slika 14 prikazuje diagram poteka za implementacijo pošiljanja sporočil s pomočjo vrste. Proces za komunikacijo BLE in proces glavne zanke si delita podatkovne strukture, vendar pa tečeta neodvisno in brez garancij za vrstni red izvajanja operacij.




*Slika 14. Diagram poteka za pošiljanje sporočil.*

Za vsako sporočilo v vrsti hranimo naslednje statuse:

- sporočilo je bilo poslano,
- sporočilo je bilo potrjeno,
- časovnik je potekel,
- sporočilo je v vrsti,
- čas, ko smo ustvarili sporočilo.

Splošno strukturo sporočil in dolžine polj v bajtih prikazuje Tabela 4. Polje vrednost je spremenljive dolžine. Tabela 5 prikazuje primer vrednosti polj za sporočilo ping.

*Tabela 4. Splošna struktura sporočila.*

| Tip | Sekvenčna številka | Čas | Vrednost |
|---|---|---|---|
| 3 B | 2 B | 4 B | 0 - 13 B |

*Tabela 5. Primer spročila ping.*

| Tip | Sekvenčna številka | Čas | Vrednost |
|---|---|---|---|
| png | 0x0001 | 0x00000021 | |

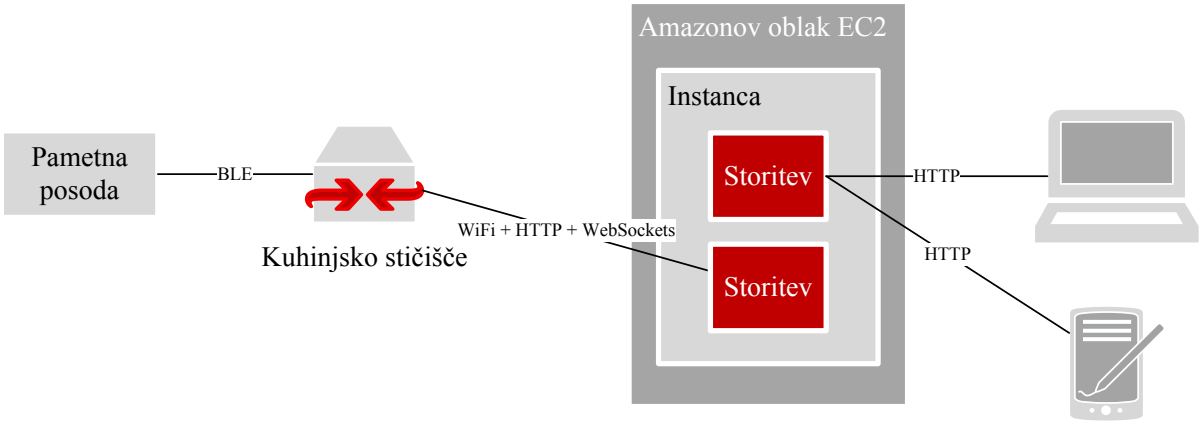
### 4.2.4. Varčevanje z energijo

Ena od zahtevanih lastnosti za pametno posodo je bila, da mora stvar v vsakem trenutku porabiti le minimalno energije. Platforma RFduino ima za to rešitev, saj jo lahko premaknemo v ultra globoko spanje v katerem potrebuje le 4 µA električnega toka. To storimo v glavni zanki kakor je prikazano na Izpis 8.

```
if (queue.IsEmpty() && leds.Idle()) {
    Simblee_ULPDelay(3000);
    //Simblee_ULPDelay(INFINITE);
}
```

*Izpis 8. Globoko spanje na RFduino.*

Tudi tukaj smo naleteli na težave. Funkcija `Simblee_ULPDelay` sprejme kot parameter število milisekund za globoko spanje ali pa konstanto `INFINITE`. Če uporabimo konstanto `INFINTE`, potem lahko našo stvar zbudimo samo še tako, da ji pošljemo sporočilo prek BLE. Vendar pa se v tem primeru nikoli več ne bo izvedla glavna zanka `loop()`, kar pomeni, da smo omejeni z viri, ki so nam na voljo znotraj prekinitvene servisne rutine (angl. interrupt service routine, ISR). Na srečo pa smo za lastnost `komunikacija.vzpostavitev` izbrali vrednost *oba*, kar pomeni da se moramo periodično zbuditi iz globokega spanca in poslati podatke posredniku.




*Slika 15. Celotna arhitektura naše aplikacije IoT*

## 4.3. Druge komponente sistema

Za voljo prikaza funkcionalnosti smo implementirali tudi preostale komponente potrebne za delovanje naše aplikacije IoT. Celotno arhitekturo in uporabljene komunikacijske protokole prikazuje Slika 15. Prehod smo poimenovali kuhinjsko stičišče, predstavljamo si, da bi lahko to vlogo v prihodnosti prevzela ena od velikih, stalno priklopljenih kuhinjskih naprav, npr. hladilnik. Trenutno predstavlja kuhinjsko stičišče skupek kode v programskem jeziku *Python*, ki teče na prenosniku z brezžičnima vmesnikoma Wi-Fi in *Bluetooth* 4.0.

Naša dodatna zahteva za kuhinjsko stičišče je bila sposobnost komunikacije iz domačega omrežja, ki je od javnega Interneta ločeno z NAT-om (angl. Network Address Translation). Za premoščanje te ovire smo uporabili protokol *WebSockets*, ki omogoča trajne povezave HTTP in HTTPS. Povezavo vedno vzpostavimo iz domačega omrežja na storitev v javnem Internetu. Ko je povezava vzpostavljena, lahko komunikacija poteka dvosmerno, brez omejitev.

### 4.3.1. Kuhinjsko stičišče

Kuhinjsko stičišče uporablja knjižnico *BluePy* [18] za komunikacijo s pametno posodo. Na drugi strani pa uporablja odjemalec *WebSocket*, ki je del ogrodja *Tornado* [29]. Izpis 9 prikazuje definicijo asinhrone funkcije, ki poišče naprave BLE v bližini kuhinjskega stičišča.

```
@gen.coroutine
def scan_devs():
    adapter = Scanner(0)
    devices = adapter.scan(10.0)
    raise gen.Return(devices)
```

*Izpis 9. Iskanje naprav BLE.*

Izpis 10 prikazuje definicijo asinhrone funkcije za povezavo z napravo BLE. Uporabo asinhronih funkcije nam omogoča ogrodje Tornado.

```
@gen.coroutine
def connect_dev(addr):
    if addr not in connections:
        connections[addr] = Peripheral(addr, 'random')
```

*Izpis 10. Povezovanje na napravo BLE.*

Izpis 11 prikazuje glavni del programa, ki vključuje povezavo na strežnik WebSocket (`websocket_connect`), branje prihajajočih sporočil (`read_message`) in pošiljanje sporočil napravi BLE (`writeCharacteristic`). Ukaz `yield` nam omogoča asinhrono proženje funkcij ne da bi pričakovanje rezultata blokiralo izvajanje programa.

```
@gen.coroutine
def main():
   client = yield tornado.websocket.websocket_connect(
            "ws://example.org:8888/ws")
    while True:
        msg = yield client.read_message()
        if msg[0:8] == 'led off ':
                addr = msg.split()[2]
                connections[addr].writeCharacteristic(17, 'off')
        ...
```

*Izpis 11. Branje sporočil iz WebSocket in pošiljanje na Bluetooth.*

Za uporabo ogrodja *Tornado* smo se odločili predvsem zaradi enostavne uporabe protokola *WebSocket*. Smo pa naleteli na težave zaradi globalnega zaklepanja tolmača (angl. *global interpreter lock*). Knjižnica *BluePy* je namreč le ovojnica okrog knjižnice *BlueZ*, ki je napisana v programskem jeziku C. Do težav prihaja, kadar uporabimo katerokoli blokirajočo funkcijo v zunanji knjižnici, takrat namreč blokira tudi tolmač *Python* in vse niti čakajo na izvedbo blokirajočega klica. V praksi to na primer pomeni, da ne moremo vzporedno brati ukazov iz naprave BLE in sprožiti iskanja novih naprav. Težavi se lahko izognemo z ustvarjanjem novih procesov za operacije BLE. Uporabimo lahko modul `multiprocessing`.

### 4.3.2. Nadzorna storitev

Nadzorno storitev smo implementirali v ogrodju *Tornado* [29]. *Tornado* je dogodkovno vodeno (angl. *event driven*) ogrodje za izdelavo spletnih storitev, še posebej primerno za vzdrževanje velikega številka hkratnih povezav *WebSocket*. Izpis 12 kaže kako vzpostavimo novo spletno

storitev z dvema točkama dostopa. Na naslovu /ws bomo zagnali strežnik *WebSocket*, naslov /led pa bo storitev REST, s katero lahko prižgemo in ugašamo diode LED na pametni posodi.

```
port = 50000
handlers = [(r"/ws", BleMasterWSHandler),
            (r"/led(On|Off|Pulse).*", LedHandler)]
app = tornado.web.Application(handlers)
app.listen(port)
tornado.ioloop.IOLoop.instance().start()
```

*Izpis 12. Vzpostavitev strežnika Tornado.*

Izpis 13 prikazuje osnovo za komunikacijo z odjemalcem *WebSocket*. V funkciji `on_message` implementiramo odgovarjanje na sporočila odjemalca. Za razliko od komunikacije v omrežju stvari, je povezava *WebSocket* prava TCP povezava, zato nam ni treba skrbeti za potrjevanje in sledenje časovnega sosledja sporočil.

```
class BleMasterWSHandler(tornado.websocket.WebSocketHandler):
    def open(self):
        self.write_message(u"ls")
    def on_message(self, message):
        ...
    def on_close(self):
        ...
```

*Izpis 13. Koncept komunikacije z odjemalcem WebSocket..*

## 4.4. Inženirjev hladilnik

Glede na Slika 16 je našem nadobudnem inženirju uspelo ukrotiti pametno posodo. Sedaj mu ne bo treba več skrbeti, da bi se mu pokvarila hrana. Zna pa se znajti v težavah, ko se bodo spraznile baterije v pametnih posodah.


*Slika 16. Inženirjev hladilnik.*

# 5. Sklepne ugotovitve

V delu smo pregledali literaturo na področju interneta stvari in na koncu definirali internet stvari kot skupino infrastruktur, ki omogočajo dostop, upravljanje in rudarjenje na podatkih, ki jih generirajo glede na okolico in interakcijo z uporabniki. Infrastrukture sestavljajo stvari, posredniki, podatkovne storitve, programski vmesniki, spletne in druge storitve, skupaj pa omogočajo prenos obdelavo in dostop podatkov s senzorjev ter nadzor aktuatorjev in spreminjanje notranjega stanja stvari. Stvari so naprave z določeno funkcijo, s senzorji za zaznavanje okolice in aktuatorji za manipulacijo okolice ter s sposobnostjo komunikacije z drugimi stvarmi, prehodi ali z drugimi omrežji.

Naša definicija nikakor ni vseobsegajoča. Deluje pa dobro v kontekstu, ki smo si ga postavili. Najbrž je to tudi eden od velikih razlogov za poplavo definicij. Praktično nemogoče je, da bi posameznik obvladoval celotno področje IoT in vse tehnologije, ki so danes del tega ekosistema, in na koncu postavil še vseobsegajočo definicijo, ki upošteva vse izjeme.

Zato se nam zdi toliko bolj pomembno, da določimo taksonomije in ontologije, ki niso popolne, so pa razširljive. Razširljivost nam v kombinaciji z odprtostjo in standardizacijo omogoči sodelovanje različnih interesnih skupin.

Prepričani smo, da je potrebno skupaj z razširljivimi modeli graditi tudi bazo skupnega znanja. V delu smo zbrali nekaj deset platform, ki smo jih uporabili v našem modelu za aplikacije IoT. Če bi vključili še deset drugih raziskovalcev, ki istočasno raziskujejo na tem področju, bi v istem času lahko prišli do več kot sto platform. Pri gradnji baze znanja si lahko zamislimo tudi sistem za testiranje modelov (angl. *model unit testing*), podobno kot imamo pri razvoju programske opreme testiranje enot (angl. *unit testing*). Vsak raziskovalec, ki razširi model, bi moral dodati tudi svoje testne primere, tako bi preizkusil, če model dejansko deluje, hkrati pa bi avtomatsko preizkusil tudi združljivost z obstoječimi podatki.

Pri postavitvi modela smo se velikokrat srečali s problemom drobljenja. Kdaj naj nehamo z dodajanjem lastnosti, ki platformo ali aplikacijo opisujejo še natančneje. Skušali smo se držati principa Occamovega rezila in na koncu model pri določenih lastnostih, npr. funkcija.interakcija, še poenostavili.

V prihodnje bi radi preizkusili pristop s strojnim učenjem. V tem primeru bi skušali aplikacije opisati s kar se da podrobnimi lastnostmi, nato pa bi uporabili npr. algoritem PCA (angl. *Principal component analysis*) za redukcijo dimenzij. Tako identificirane dimenzije bi primerjali z lastnostmi, ki smo jih določili ročno.

Podoben pristop želimo preizkusiti tudi pri identifikaciji najprimernejših platform za dane zahteve aplikacije IoT. Če zberemo dovolj podatkov o platformah, lahko uporabimo algoritme za klasifikacijo v več razredov (angl. *multi-label classification*). Na ta način bi dobili tudi neko oceno primernosti, s pomočjo katere bi lahko rangirali priporočila tako za platformo, kot za arhitekturo.

Izdelano orodje je že danes primerno za pomoč inženirjem, ki vstopajo v svet IoT, saj lahko pomembno zmanjša čas za odločitev o platformi za realizacijo aplikacije IoT.

Presenetilo nas je, da smo lahko s preudarno postavljenim modelom in le nekaj statično določenimi pravili določili arhitekturo aplikacije IoT.

Z izdelavo aplikacije smo pokazali, da je naš predolg modela aplikacije IoT izvedljiv. Pokazali smo, da tudi zelo preproste aplikacije IoT vključujejo različne storitev in nabor tehnologij, vsako s svojim naborom kompleksnosti. Vsekakor pozdravljamo prihajajoča ogrodja, s katerimi zgradimo en model aplikacije, ogrodje pa ga samo razdeli na komponente modela aplikacije IoT.

# Viri

[1] I. Alqassem and D. Svetinovic, "A taxonomy of security and privacy requirements for the Internet of Things (IoT)," *2014 IEEE Int. Conf. Ind. Eng. Eng. Manag.*, pp. 1244–1248, 2014.

[2] E. Aractingi, "Towards a common definition and taxonomy of the Internet of Things Contents," *Internet2, CINO Work. Groups, Internet Things*, pp. 1–9, 2015.

[3] K. Ashton, "That 'internet of things' thing," *RFiD J.*, 2009.

[4] L. Barker, M. White, M. Curran, and B. Huggin, "Taxonomy for Internet of Things: Tools for monitoring Personal Effects," *Int. Conf. Pervasive Embed. Comput. Commun. Syst. (PECCS 2014)*, 2014.

[5] J. P. Conti, M. Chui, M. Loeffler, N. Gershenfeld, R. Krikorian, and D. Cohen, "The Internet of things," *Commun. Eng.*, vol. 4, no. 4, pp. 76–81, 2004.

[6] B. Dorsemaine, J. P. Gaulier, J. P. Wary, N. Kheir, and P. Urien, "Internet of Things: A Definition and Taxonomy," *Proc. - NGMAST 2015 9th Int. Conf. Next Gener. Mob. Appl. Serv. Technol.*, pp. 72–77, 2016.

[7] P. Harrop, J. Harrop, and D. Raghu, *Internet of Things (IoT): Business Opportunities 2015-2025*. IDTechEx, 2015.

[8] M. Loukides and J. Bruner, *What is the Internet of things*. O'Reilly, 2015.

[9] D. Lund and M. Morales, "Worldwide and Regional Internet of Things ( IoT ) 2014 – 2020 Forecast : A Virtuous Circle of Proven Value and Demand," no. May, pp. 2014–2020, 2014.

[10] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon, "The internet of things: mapping the value beyond the hype," 2015.

[11] V. Madisetti, A. Bahga, Internet of Things, First Edition, VPT, 2014.

[12] R. Stackowiak, A. Licht, V. Mantha, L. Nagode, Big Data and the Internet of Things: Enterprise Information Architecture for a New Age, Apress, 2015.

[13] C. Swedberg, "Hong Kong Airport Says It Now Uses Only RFID Baggage Tags Hong Kong Airport Says It Now Uses Only RFID Baggage Tags," *RFiD J.*, pp. 1–2, 2009.

[14] L. Tan, N. Wang, "Future internet: The Internet of Things," Advanced Computer Theory and Engineering, 2010 3rd International Conference on, Chengdu, 2010.

[15] O. Vermesan, et al., Internet of Things - Strategic Research Roadmap, European Commission-Information Society and Media DG. Brussels, 2009.

[16] T. Zhang, Y. Ouyang, and Y. He, "Traceable air baggage handling system based on RFID tags in the airport," *J. Theor. Appl. Electron. Commer. Res.*, vol. 3, no. 1, pp. 106–115, 2008.

[30] J. Lee, Y. Su, and C. Shen, "A Comparative Study of Wireless Protocols :," *IECON Proc. (Industrial Electron. Conf.*, pp. 46–51, 2007.

[31] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4," *2012 IEEE Wirel. Commun. Netw. Conf. Work. WCNCW 2012*, pp. 232–237, 2012.

[32] K. Mikhaylov, N. Plevritakis, and J. Tervonen, "Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI," *J. Sens. Actuator Networks*, vol. 2, no. 3, pp. 589–613, 2013.

[17] (2016) Amazon Web Services IoT. Dostopno na: https://aws.amazon.com/iot/

[18] (2016) BluePy. Dostopno na: https://github.com/IanHarvey/bluepy

[19] (2016) Fitbit One. Dostopno na: https://www.fitbit.com/one

[20] (2016) IT Glossary, Internet of Things. Dostopno na: http://www.gartner.com/it-glossary/?s=internet+of+things

[21] (2016) Gartner Says 6.4 Billion Connected "Things" Will Be in Use in 2016. Dostopno na: http://www.gartner.com/newsroom/id/3165317

[22] (2016) IBM Bluemix Internet of Things. Dostopno na: http://www.ibm.com/cloud-computing/bluemix/internet-of-things/

[23] (2016) Navigating IoT through the hype, Connecting the EDGE, Internet of Things Applications USA, IDTechEx. Dostopno na: http://www.idtechex.com/internet-of-things-usa/show/en/

[24] (2016) Microsoft Azure IoT Suite. Dostopno na: https://www.microsoft.com/en-us/cloud-platform/internet-of-things-azure-iot-suite

[25] (2016) MQTT protocol. Dostopno na: http://mqtt.org/

[26] (2016) J. Morgan, A Simple Explanation Of 'The Internet Of Things', 2014. Dostopno na: http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/

[27] (2016) PCH, we make. Dostopno na: http://www.pchintl.com/

[28] (2016) The Thing System. Dostopno na: http://thethingsystem.com/

[29] (2016) Tornado. Dostopno na: http://www.tornadoweb.org/en/stable/

[33] (2016) IoT Hardware Guide. Dostopno na: http://www.postscapes.com/internet-of-things-hardware/